HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

THESIS

Unsupervised Domain Adaptation for Event Detection

Author:
Nghia NGO TRUNG

Supervisor:
M.Sc. Linh NGO VAN

Department of Information Systems

July 19, 2021

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Abstract

School of Information and Communication Technology
Department of Information Systems

Unsupervised Domain Adaptation for Event Detection

by Nghia NGO TRUNG

Machine learning algorithms are procedures that train an mathematical model to extract informative statistics from observed data for making predictions or regression on another unseen dataset. In recent years, the explosion of deep learning methods and powerful models have significantly pushed forward the potential of artificial learners, achieving remarkable state-of-the-art performances on every benchmarks of all fields. Despite the extraordinary success of the modern algorithms, a large majority of them still rely on a standard assumption of classical approaches to work: training and testing data follow the same distribution and the trained models can extract useful knowledge from train dataset and leverage them to perform generally well on unknown test dataset. This simplification leads to the common overfitting phenomenon where the learned models are unable to adapt to different tasks or even the same tasks with data coming from different domains. In practice, this is usually the case for most applications. Thus, motivated by how humans perceive and process information, another learning setting is formulated to address the problem: Transfer Learning. The key idea of transfer learning is to exploit accessible data from one or many areas and using them to improve capability of models for related tasks. One of the hardest formulation of it is unsupervised domain adaptation, where the model is trained on a labelled source domain dataset while also having access to a small unlabelled set of unlablled samples drawn from another domain. This domain, refer to as target domain, is the domain on which the model is tested and may be substantially different from the source domain, therefore causing significantly decrease in model's performance. Many approaches have been proposed for this particular setting, most of which however were developed and tested on a controlled environment with simple datasets and small models. To adopt these methods on practical applications is still a challenging problem, especially in the field of natural language processing where large language models are essential for achieving good performance. In this thesis, we propose a solution for the above issue. Our model, Adapter-based Hierarchical Domain Separation network (AHDS), leverages a special finetuning procedure using adapter modules to efficiently make use of the capacity of the enormous language models. Furthermore, we devise a novel mechanism to address the discrepancy between domains in hierarchical fashion, which consists of a data selection mechanism and a distribution alignment mechanism. Experimental results and extensive ablation studies on event-related classification tasks demonstrate the effectiveness of our approach in comparison to other state-of-the-art methods.

Acknowledgements

This thesis would not have been possible without the help of the following people, all of whom I would like to thank here.
First of all, I would like to express my sincere gratitude to my supervisor M.Sc. Ngo Van Linh for his counseling throughout the process. Without his encouragement and suggestion, this thesis could never have been completed. Furthermore, I also want to send my appreciation to my mentor Prof. Nguyen Huu Thien, from whom I have received countless invaluable insights and reviews on the subject. I am more than grateful to receive the unconditional help and guidance from them.
I am grateful to Hanoi University of Science and Technology (HUST) and to all lectured who guided me through various lessons different areas of computer science. I also want to express my thanks to all the organizers of ICT, School of Information and Communication Technology (SoICT), for making it possible for me to meet and study together with my classmates, who have influenced and inspired me greatly during the learning process. I am humble and thankful for the knowledge I have learnt from each and all of you.
Finally, to my family whose encouragement acts as a pillar of support for me no matter the circumstances. Thank you for always staying by me, providing comfort while also pushing me forward so I can overcome any obstacle on the way. I am forever thankful for it.

Contents

List of Figures

List of Tables

List of Abbreviations

AHDS	**A**dapter-based **H**ierarchical **S**eparation
AbF	**A**dapter based **F**inetuning
BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
CV	**C**omputer **V**ision
DAPT	**D**omain **A**daptive **P**re **T**raining
DANN	**D**omain **A**dversarial **N**eural **N**etwork
DSN	**D**omain **Separation** **N**etwork
EE	**E**vent **E**xtraction
EI	**E**vent **I**dentification
ED	**E**vent **D**etection
LM	**L**anguage **M**odel
IPM	**I**ntegral **P**robability **M**etric
GAN	**G**enerative **A**dversarial **N**etwork
GRL	**G**radient **R**eversal **L**ayer
MLM	**M**asked **L**anguage **M**odel
NLP	**N**atural **L**anguage **P**rocessing
SDA	**S**upervised **D**omain **A**daptation
SSDA	**S**emi **S**upervised **D**omain **A**daptation
UDA	**U**nsupervised **D**omain **A**daptation
RNN	**R**ecurrent **N**eural **N**etwork
TAPT	**T**ask **A**daptive **P**re **T**raining

Chapter 1

Introduction

1.1 Problems

Natural Language Processing (NLP) is one of the main field of artificial intelligence research and plays an importance role on the quest towards understanding general intelligence. As language is ubiquitous in almost every human interactions, the ability to process and extract informative knowledge from the arbitrary text is of utmost necessity to any natural language understanding system, especially in current modern era where both the amount and the accessibility of uncurated data have significantly increased. To provide model with this ability, NLP researcher have formulated it in the task of Event Extraction (EE), an extremely complicated task which further involves two sub-tasks: Event Identification (EI) and Event Detection (ED). The former is a low-level task where one needs to locate the word that trigger an event in a given context, while the latter is a high-level task in which the trigger is also required to be classified into a set of event types. These sub-tasks can be formulated simply as text classification problems, where ones' job is to label the concerning text into one of the predefined categories. However, complication happens when datasets are collected from different domains, thus undergo some types of shift. These changes can potentially affect performance if they are not addressed by the learning algorithm. For examples, in event identification task, what decide a trigger word to be an event is heavily depended on the corpus from which the word is drawn. An event in formal situation may gets ignored by annotators if they get instruct to focus more on casual context. Likewise, the meaning of a trigger can change according to the context to which its belong. As a result, the same word may have two different labels moving from one corpus to another.

Classification task can be solved effectively by machine learning algorithms that train a mathematical model to extract informative statistics from observed data to make predictions on another unseen dataset. However, the majority of these methods still rely on the basic supervised learning assumption where training and testing data follow the same distribution, thus unable to adapt to more difficult data settings such as domain adaptation. Recently, deep learning has pushed the boundary of what an artificial learner can understand, especially the unsupervised learning approaches using large capacity models. For the field of NLP in particular, self-attention based language models are able to learn and represent linguistic context of data, significantly enhance the discriminative power of text representations. These models outperform previous approaches on every NLP benchmarks by a wide margin, even in difficult tasks with data coming from various domains. Nevertheless, a finetuning procedure is needed to refine the general representations into more informative features for downstream tasks, in which the entire model is retrained. Because of the excessive size of the models, this is impractical for most applications

where multiple adaptations are needed. Thus, there is a need for a better framework that can effectively leverage the ability of pretrained models for general learning setting while also keeping the concern of efficiency in mind.

1.2 Goals and Scopes

Inspired by the desiderata laid out by [Rud19] for the general transfer learning setting, we suggest the following key properties that a model should have for the task of unsupervised domain adaptation:

- **Domain Adaptation**: The model should explicit minimize the discrepancy in data between the source and target domains, regardless of how distinct the two domains are to each other.
- **Architecture Augmentation**: The model should be able to leverage the abilities of other unsupervised pretrained models, which have become the current state-of-the-art for most tasks in supervised setting.
- **Generalization Performance**: The model should be designed with the ability to generalize as the central goal. Certain types of inductive biases should be included to drive the model toward that goal.
- **Computational Efficiency**: The model should be computational efficiency so as to adapt to the situation where there are more available data from multiple domains. Furthermore, naive re-training for each domain approach should be impractical if the framework makes use of large pretrained models. Thus, the trade-off between efficiency and performance should be taken into serious consideration.
- **Task-agnostic and Domain-agnostic**: The model should be applicable to various different settings. It should be effective across wide range of domains (different languages, syntactic and semantic) and tasks (low-level tasks, i.e. identification, to high-level task, i.e. classification, translation), without or with minimal changes to the overall architecture.

Suggested Solutions Taking all of the above factors into consideration, we propose a novel framework for the task of text classification in unsupervised domain adaptation setting: Adapter-based Hierarchical Domain Separation network (AHDS). Our method follows the adapter-based finetuning framework, thus able to make use of large pretrained LM in efficient way, while also using multiple approaches to address the problem of domains shift and learning domain-invariant features. Experiments with different settings together with extensive analyses have empirically demonstrated the usefulness of our approach.

1.3 Thesis Structure

The main contents of this thesis will be structured as followed:

- **Chapter 2**: We provide on overview of background information regarding domain adaptation problem in the context of machine learning. We review some important seminal results for the learning setting and deep learning methods that are motivated by them.

- **Chapter 3**: We go into the specific field of NLP, presenting the current state-of-the-art language models, along with several previous approaches suggested for improving their transfer learning ability.
- **Chapter 4**: We propose a novel architecture: Adapter-based Hierarchical Domain Separation network. Our approach addresses the unsupervised domain adaptation problem for the task of text classification by first select an appropriate subset of source instances, then use them in conjunction with the unlabelled target samples to extract domain-invariant features. Finally, these features are used as inputs for the classification task on source dataset.
- **Chapter 5**: We present the experimental results of our framework on two event-related tasks: event identification and event detection. Our approach consistently outperforms both baselines and current state-of-the-art model . In addition, we provide extensive ablation studies and analyses to identify the contributions of our proposed components.
- **Chapter 6**: We conclude the thesis, summarize our findings and point out possible directions for the future research of our work.

Chapter 2

Backgrounds

This chapter first present an outline of transfer learning setting in general, which include its motivation and problem definition (§ 2.1). We then look into our main concern - domain adaptation - a sub-field of transfer learning. Detail survey is provided about the various contexts in which it is involved and its categorization (§ 2.2). Finally, we introduction two popular approaches for the problem of unsupervised domain adaptation, both having the same goal which is distribution alignment between source and target domain (§ 2.3).

2.1 Transfer Learning

2.1.1 Motivations

Machine learning algorithms in general are procedures that train an mathematical model to extract informative statistics from observed data for making predictions or regression on another unseen dataset. While this setting can result in very different problems in various situations, a large majority of algorithms still rely on a standard assumption to work: training and testing data follow the same distribution and the trained models can extract useful knowledge from train dataset and leverage them to perform generally well on unknown test dataset. However, this simplification also leads to the common overfitting phenomenon where the learned models are unable to adapt to different tasks or even the same tasks with data coming from different domains. In practice, this is usually the case for most applications. Thus, another learning setting is formulated to address the problem: Transfer Learning.

Inspired by the fact that a human can extrapolate information across tasks to generalize and learn new task more efficiently, the key idea of transfer learning is to exploit accessible data from one or many areas and using them to improve capability of models for related tasks. A transfer learning model has many desirable properties, from the ability to generalize to various different tasks, to being robust to the noise or unimportant information from within the same task.

2.1.2 Problem Definitions

Transfer learning Consider data drawing i.i.d from two distribution $\mathcal{S}$ and $\mathcal{T}$, which we refer to as source domain and target domain correspondingly. Let $\mathbf{C}_\mathcal{S}$ and $\mathbf{C}_\mathcal{T}$ be the source and target learning tasks, each defines a labeling function $f_i : \mathbf{X}_i \rightarrow \mathbf{Y}_i$ mapping the domain's input space $\mathbf{X}_i$ into the associated output space $\mathbf{y}_i$ ($i \in \{\mathcal{S}, \mathcal{T}\}$). Then, the goal of transfer learning model is to find an optimal function $h_\mathcal{T}$ that approximate $f_\mathcal{T}$ as close as possible, using knowledge extracted from the (fully known) $\mathcal{S}$ and the (partially known or completely unknown) $\mathcal{T}$ where $\mathcal{S} \neq \mathcal{T}$.

Remark For brevity, we only consider the setting where there is one source domain and one target domain. It is simply to generalize the formulation into the case where multiple source domains are accessible (refer to [Zha+20] for further detail)

2.1.3 Transfer Settings

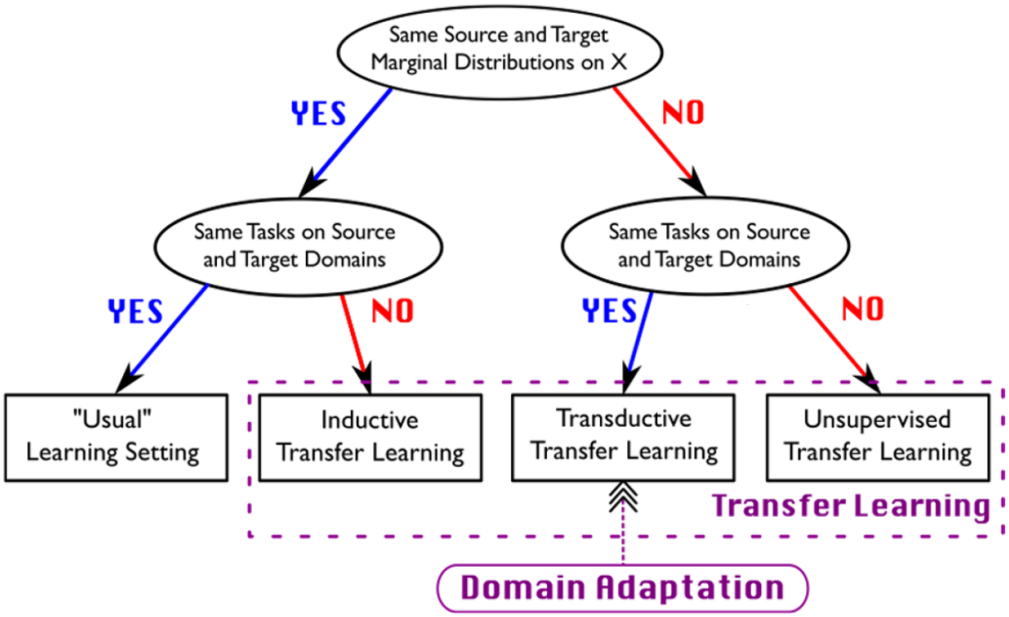


FIGURE 2.1: Transfer Learning Categories (from [Red+19]).

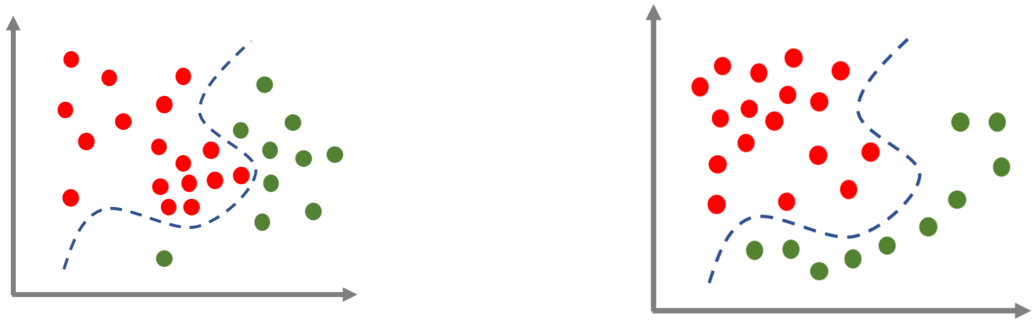
Based on the relations between the joint data space $\mathbf{X}_i \times \mathbf{Y}_i$ of domains, we can further categorized transfer learning into three possible learning settings:

Inductive Transfer Learning corresponds to situation where both domains share similar marginal distribution $\mathbf{P}_{\mathcal{S}}(\mathbf{X}) = \mathbf{P}_{\mathcal{T}}(\mathbf{X})$ but aim at different tasks $\mathbf{C}_{\mathcal{S}} \neq \mathbf{C}_{\mathcal{T}}$. For example, the same news corpus can serve as dataset for multiple tasks such as entity recognition, event detection, ...

Transductive Transfer Learning corresponds to situation where each domain have a different marginal distribution $\mathbf{P}_{\mathcal{S}}(\mathbf{X}) \neq \mathbf{P}_{\mathcal{T}}(\mathbf{X})$ but share the same task $\mathbf{C}_{\mathcal{S}} = \mathbf{C}_{\mathcal{T}}$. For example, given the task of question answering, dataset from unrelated corpora can have different types of questions and contexts.

Unsupervised Transfer Learning corresponds to situation where marginal distributions and tasks vary across domains, $\mathbf{P}_{\mathcal{S}}(\mathbf{X}) \neq \mathbf{P}_{\mathcal{T}}(\mathbf{X})$ and $\mathbf{C}_{\mathcal{S}} \neq \mathbf{C}_{\mathcal{T}}$. An exemplary case is transferring knowledge from unsupervised learning of large general-domain corpora to specific downstream tasks.

The most popular of the three is transductive transfer learning, also refer to as domain adaptation, the main focus of this thesis. In domain adaptation, data is gather from two different origins for the same given task. One is the labelled source dataset with abundant amount of samples, while the other is the target dataset of moderate-sized sample collection but little to none labelled instances.



(A) Source Domain (B) Target Domain

FIGURE 2.2: Covariate Shift Settings (from [LP20])

2.2 Domain Adaptation

2.2.1 Types of Dataset Shift

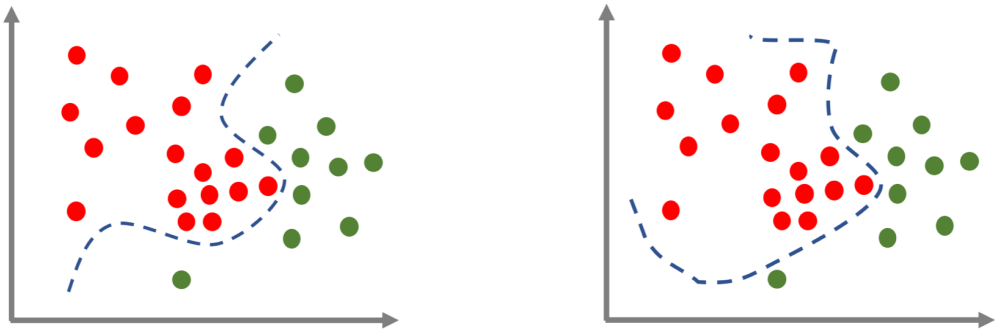
Dataset shift is term used to describe the situation when there is a change in the joint distribution $\mathbf{P}(\mathbf{X}, \mathbf{Y})$ moving from source domain to target domain, which usually worsen the target performance of naive models that simply being trained on only source dataset. [KF14] has studied many different sources of the phenomenon and proposed to systematical group them based the factorization of the joint distribution as follow:

- **Covariate Shift** the shift between domain is fully accounted for by the marginal distribution of input, which means $\mathbf{P}_{\mathcal{S}}(\mathbf{X}) \neq \mathbf{P}_{\mathcal{T}}(\mathbf{X})$ while the predictive distributions on labels given the inputs stay the same $\mathbf{P}_{\mathcal{S}}(\mathbf{Y}|\mathbf{X}) = \mathbf{P}_{\mathcal{T}}(\mathbf{Y}|\mathbf{X})$
- **Target Shift** occurs when the prior distributions of labels are different between domains $\mathbf{P}_{\mathcal{S}}(\mathbf{Y}) \neq \mathbf{P}_{\mathcal{T}}(\mathbf{Y})$. However, the class conditional distributions remain unchanged $\mathbf{P}_{\mathcal{S}}(\mathbf{X}|\mathbf{Y}) = \mathbf{P}_{\mathcal{T}}(\mathbf{X}|\mathbf{Y})$
- **Conditional Covariate Shift** the case when different domains define distinct dependencies of input $\mathbf{X}$ on output $\mathbf{Y}$, in other word $\mathbf{P}_{\mathcal{S}}(\mathbf{X}|\mathbf{Y}) \neq \mathbf{P}_{\mathcal{T}}(\mathbf{X}|\mathbf{Y})$. In contrast, the marginal distributions are assumed to be invariant across domains $\mathbf{P}_{\mathcal{S}}(\mathbf{X}) = \mathbf{P}_{\mathcal{T}}(\mathbf{X})$
- **Concept Shift** refer to the situation in which domains' labeling functions define different conditional distributions, or $\mathbf{P}_{\mathcal{S}}(\mathbf{Y}|\mathbf{X}) \neq \mathbf{P}_{\mathcal{T}}(\mathbf{Y}|\mathbf{X})$, based on the same marginal distributions $\mathbf{P}_{\mathcal{S}}(\mathbf{X}) = \mathbf{P}_{\mathcal{T}}(\mathbf{X})$

2.2.2 Learning Settings

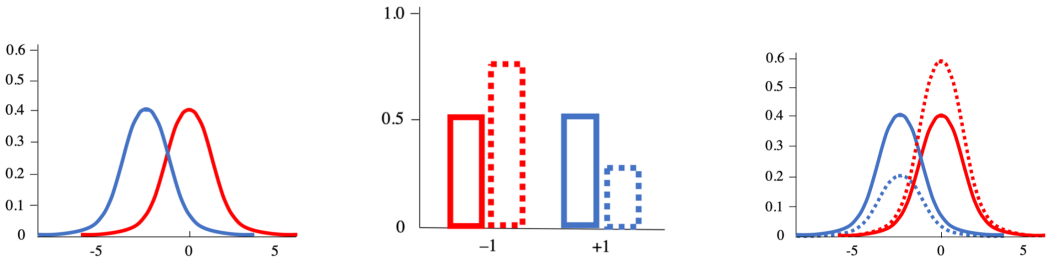
An algorithm addressing domain adaptation follow the same learning setting as other machine learning algorithms, with some modification to the training dataset to reduce the discrepancy between domains

Supervised Domain Adaptation (SDA) In supervised setting, a small amount of labeled data is available from target domain $\mathcal{T}$, along with a larger dataset of labeled data from source domain $\mathcal{S}$. This is the easiest setting where The goal is to using both datasets to train a model optimized for target domain



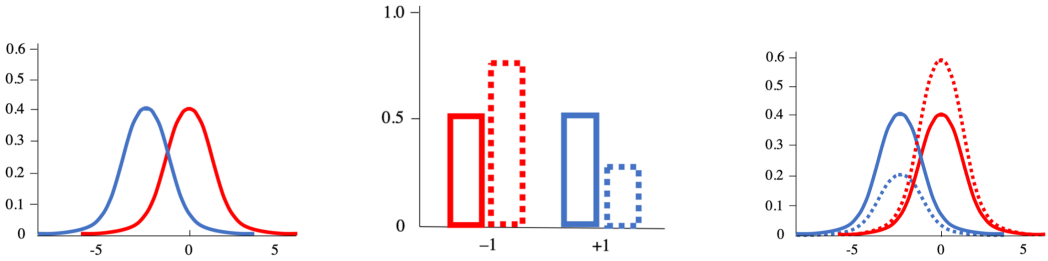
(A) Source Domain (B) Target Domain

FIGURE 2.3: Concept Shift Settings (from [LP20]).



(A) P(X) (B) P(Y | X) (C) P(XY)

FIGURE 2.4: Conditional Covariate Shift Settings (from [LP20]).



(A) P(X | Y) (B) P(Y) (C) P(XY)

FIGURE 2.5: Target Shift Settings (from [LP20]).

Semi-Supervised Domain Adaptation (SSDA) In semi-supervised setting, the amount of accessible labeled target data is kept to be minimum compared to SDA. This is a more realistic scenario in which the key question is how to maximally make use of the labeled target data to enhance model's capability for target domain.

Unsupervised Domain Adaptation (UDA) In unsupervised setting, model has to learn from unlabeled target data without access to any information regarding labeling function of target domain. This is the hardest setting of the three and the only possible optimal solution is probably extracting general statistics that are shared across domains, thus can be used for improving target domain's performance of the model.

2.3 Distribution Alignment Methods

We only present here the bare minimum of the required definitions and bounds here, which motivate the following approaches. For more detail information regarding their full mathematical concepts and proofs, please refer to comprehensive survey of the corresponding subjects ([Red+19])

2.3.1 Divergence-based Approaches

In UDA setting, it is intuitive to make an assumption that a model that is trained to bring marginal source distribution $\mathbf{P}_{\mathcal{S}}(\mathbf{X})$ closer to $\mathbf{P}_{\mathcal{T}}(\mathbf{X})$ would also be able to transfer its predictive ability from source domain to target domain. However, finding a suitable distance between the two domains or a divergence to quantify the difference of their corresponding distributions is a complicate problem that require many considerations. Initial research on domain adaption [Ben+07] proposed to use L^1-distance to bound the difference between $\mathcal{S}$ and $\mathcal{T}$

Hypothesis Difference Given two hypothesises h and h' in $\mathcal{H}$. The probability according to the distribution $\mathbf{P}_{\mathcal{S}}(\mathbf{X})$ that h disagrees with h' based on loss function ℓ define as:

$$\mathcal{E}^{\ell}_{\mathbf{x}\sim\mathcal{S}}\left(h, h'\right) = \mathbf{E}_{\mathbf{x}\sim\mathcal{S}}\left[\ell(h(\mathbf{x}) - h'(\mathbf{x}))\right].$$

We use shorthand $\mathcal{E}^{\ell}_{\mathbf{x}\sim\mathcal{S}}(h) = \mathcal{E}^{\ell}_{\mathbf{x}\sim\mathcal{S}}(h, f_{\mathcal{S}})$ to refer to generalization source error, and $\widehat{\mathcal{E}^{\ell}_{\mathbf{x}\sim\mathcal{S}}(h)}$ for empirical source error. For target domain, parallel notations are used $\mathcal{E}^{\ell}_{\mathbf{x}\sim\mathcal{T}}(h)$ and $\widehat{\mathcal{E}^{\ell}_{\mathbf{x}\sim\mathcal{T}}(h)}$.

L^1-distance If ℓ is the L^1 norm, then $\mathrm{disc}_{\ell}\left(\mathbf{P}_{\mathcal{S}}(\mathbf{X}), \mathbf{P}_{\mathcal{T}}(\mathbf{X})\right)$ becomes the L^1-distance or the total variation distance, which is a proper metric on the space of probability distributions that measures the largest possible difference between $\mathbf{P}_{\mathcal{S}}(\mathbf{X})$ and $\mathbf{P}_{\mathcal{T}}(\mathbf{X})$

$$\mathrm{disc}_{L1}\left(\mathbf{P}_{\mathcal{S}}(\mathbf{X}), \mathbf{P}_{\mathcal{T}}(\mathbf{X})\right) = 2 \sup_{\mathbf{x}\sim\mathcal{S}} \left|\mathbf{P}_{\mathcal{S}}(\mathbf{x}) - \mathbf{P}_{\mathcal{T}}(\mathbf{x})\right| \tag{2.1}$$

L^1-Bound Motivated by the popularity of the total variation distance in many real practical applications,the first importance result on domain adaptation tried to bound the gap between the source and target domains' 0-1 loss functions using L^1-distance

$$\forall h \in \mathcal{H},\ \mathcal{E}^{\ell_{01}}_{\mathbf{x}\sim\mathcal{T}}(h) \leq \mathcal{E}^{\ell_{01}}_{\mathbf{x}\sim\mathcal{S}}(h) + \mathrm{disc}_{L1}\left(\mathbf{P}_{\mathcal{S}}(\mathbf{X}), \mathbf{P}_{\mathcal{T}}(\mathbf{X})\right) + \lambda_{\mathcal{ST}} \tag{2.2}$$

The third term on the right-hand side of bound quantifies the amount of disagreement between labeling functions of $\mathcal{S}$ and $\mathcal{T}$, thus cannot be controlled by a learning algorithm. The key idea is by optimizing the classifier h for source domain $\mathcal{S}$, the bound **??** is expected to also push the loss with respect to target domain $\mathcal{T}$ lower, which effectively means knowledge learned from $\mathcal{S}$ has been transfer to $\mathcal{T}$.

Despite the theoretical results, Using L^1-distance imposes two significant restrictions that make its usage impractical for the learning task. First, the L^1-distance cannot be accurately approximated by learning from finte samples for every probability distributions. Another problem is that there is no dependence on the considered hypothesis h from the definition of L^1-distance. As a result, the bound **??** can be very loose inequality and lead to a sub-optimal hypothesis for target domain. In order to address these issues, following works [Ben+10] further proposed to use a better distance for the bound, which went under the name of $\mathcal{H}\Delta\mathcal{H}$-divergence.

Discrepancy distance Consider two domains $\mathcal{S}$ and $\mathcal{T}$ over joint space $\mathbf{X} \times \mathbf{Y}$, classifier $h : \mathbf{X} \to \mathbf{Y}$ in hypothesis class $\mathcal{H}$ and loss function $\ell : \mathbf{Y} \times \mathbf{Y} \to \mathbb{R}_+$. The discrepancy distance $disc_\ell$ between two marginal distributions $\mathbf{P}_\mathcal{S}(\mathbf{X})$ and $\mathbf{P}_\mathcal{T}(\mathbf{X})$ is defined by

$$\mathrm{disc}_\ell\left(\mathbf{P}_\mathcal{S}(\mathbf{X}), \mathbf{P}_\mathcal{T}(\mathbf{X})\right) = \sup_{(h,h')\in\mathcal{H}^2} \left|\mathbf{E}_{\mathbf{x}\sim\mathcal{S}}\left[\ell\left(h'(\mathbf{x}), h(\mathbf{x})\right)\right] - \mathbf{E}_{\mathbf{x}\sim\mathcal{T}}\left[\ell\left(h'(\mathbf{x}), h(\mathbf{x})\right)\right]\right| \tag{2.3}$$

$\mathcal{H}\Delta\mathcal{H}$-divergence Let $(g, g')^2 \in \mathcal{H}^2$. Denote $\mathcal{H}\Delta\mathcal{H}$ the symmetric difference hypothesis space define as

$$h \in \mathcal{H}\Delta\mathcal{H} \iff h(\mathbf{x}) = g(\mathbf{x}) \oplus g'(\mathbf{x}) \tag{2.4}$$

where $\oplus$ stands for XOR operation. Then the $\mathcal{H}\Delta\mathcal{H} - divergence$ between $\mathbf{P}_\mathcal{S}(\mathbf{X})$ and $\mathbf{P}_\mathcal{T}(\mathbf{X})$ is defined as:

$$\begin{aligned}
&\mathrm{disc}_{\ell_{\mathcal{H}\Delta\mathcal{H}}}\left(\mathbf{P}_\mathcal{S}(\mathbf{X}), \mathbf{P}_\mathcal{T}(\mathbf{X})\right) \\
&= 2 \sup_{h\in\mathcal{H}\Delta\mathcal{H}} \left|\mathbf{P}_\mathcal{S}(I(h)) - \mathbf{P}_\mathcal{T}(I(h))\right| && (2.5)\\
&= 2 \sup_{(g,g')\in\mathcal{H}^2} \left|\mathbf{E}_{\mathbf{x}\sim\mathcal{S}}\left[\ell_{01}\left(g(x), g'(x)\right)\right] - \mathbf{E}_{\mathbf{x}\sim\mathcal{T}}\left[\ell_{01}\left(g(x), g'(x)\right)\right]\right| && (2.6)\\
&= 2\ \mathrm{disc}_{\ell_{01}}\left(\mathbf{P}_\mathcal{S}(\mathbf{X}), \mathbf{P}_\mathcal{T}(\mathbf{X})\right) && (2.7)
\end{aligned}$$

where $I(h)$ is the set for which h is the characteristic function, that is $\mathbf{x} \in I(h) \Leftrightarrow h(\mathbf{x}) = 1$.

By definition, for all class $\mathcal{H}$ the $\mathcal{H}\Delta\mathcal{H}$-divergence is never larger than the L^1-distance. Furthermore, using statistical learning theory and VC dimension, it is possible to estimate the $\mathcal{H}\Delta\mathcal{H}$-divergence from finite samples

$$\mathrm{disc}_{\ell_{\mathcal{H}\Delta\mathcal{H}}}\left(\mathbf{P}_\mathcal{S}(\mathbf{X}), \mathbf{P}_\mathcal{T}(\mathbf{X})\right) \leq \widehat{\mathrm{disc}}_{\ell_{\mathcal{H}\Delta\mathcal{H}}}\left(\mathbf{P}_\mathcal{S}(\mathbf{X}), \mathbf{P}_\mathcal{T}(\mathbf{X})\right) + \mathbf{C}\left(m, VC(\mathcal{H})\right) \tag{2.8}$$

where the second right-hand side term depends on the VC dimension of $\mathcal{H}$ and goes to zero as number of samples m increase.

$\mathcal{H}\Delta\mathcal{H}$-Bound Using above inequalities together with the previous L^1-Bound, we are able to derived a tighter bound for the generalized loss for target domain using the empirical $\mathcal{H}\Delta\mathcal{H}$-divergence between the marginal distributions $\mathbf{P}_{\mathcal{S}}(\mathbf{X})$ and $\mathbf{P}_{\mathcal{T}}(\mathbf{X})$

$$\forall h \in \mathcal{H},\ \mathcal{E}^{\ell_{01}}_{\mathbf{x}\sim\mathcal{T}}(h) \leq \mathcal{E}^{\ell_{01}}_{\mathbf{x}\sim\mathcal{S}}(h) + \widehat{\text{disc}}_{\ell_{\mathcal{H}\Delta\mathcal{H}}}(\mathbf{P}_{\mathcal{S}}(\mathbf{X}), \mathbf{P}_{\mathcal{T}}(\mathbf{X})) + \lambda^*_h + \mathbf{C}(m, VC(\mathcal{H})) \tag{2.9}$$

where λ^*_h is the combined risk of the ideal hypothesis h^* which minimize $\mathcal{E}^{\ell_{01}}_{\mathbf{x}\sim\mathcal{T}}(h) + \mathcal{E}^{\ell_{01}}_{\mathbf{x}\sim\mathcal{S}}(h)$. While this term depend on the considered hypothesis h, it is impossible to estimate its value because of the need for target label in its expression. However, as it is the risk of optimal h, its contribution compared to other terms should be negligible in most case. Thus, by finding a classifier h that minimizes both the empirical source error and the approximated $\mathcal{H}\Delta\mathcal{H}$-divergence between $\mathcal{S}$ and $\mathcal{T}$, we can achieve decent performance on target domain.

Domain-Adversarial Neural Network (DANN) To adopt above results to learning procedure of a neural network is not a trivial matter and requires the usage of generative adversarial network (GAN), which is the state-of-the-art approach for black-box density estimation from the field of deep learning at the time. In essence, DANN ([Gan+16]) proposed to extract domain-invariant features from only unlabelled data of both domain through an additional domain classifier h_{dom} (aside from the main task classifier h_{task}. The goal is that the representations learned by the neural network, while discriminative for the task at hand, are also indistinguishable to h_{dom}. DANN achieves this by pushing the feature extractor ϕ of the network to both minimize task loss through h_{task} and maximally misdirect h_{dom}, while also minimizing the domain-classification loss through h_{dom}. This min-max optimization procedure stems from the following identity of the empirical $\mathcal{H}\Delta\mathcal{H}$-divergence

$$\begin{aligned}
&\widehat{\text{disc}}_{\ell_{\mathcal{H}\Delta\mathcal{H}}}(\mathbf{P}_{\mathcal{S}}(\mathbf{X}), \mathbf{P}_{\mathcal{T}}(\mathbf{X})) \\
&= 2\left(1 - \min_{h\in\mathcal{H}\Delta\mathcal{H}}\left[\frac{1}{m}\sum_{\mathbf{x}:h(\mathbf{x})=0}\mathbf{I}[\mathbf{x}\sim\mathbf{P}_{\mathcal{S}}(\mathbf{X})] + \frac{1}{m}\sum_{\mathbf{x}:h(\mathbf{x})=1}\mathbf{I}[\mathbf{x}\sim\mathbf{P}_{\mathcal{T}}(\mathbf{X})]\right]\right) &(2.10)\\
&\approx 2\left(1 - \min_{h\in\mathcal{H}\Delta\mathcal{H}}\widehat{\mathcal{E}}^{\ell}_{dom}(h_{dom}(\phi(\mathbf{x}), f_{dom})\right) &(2.11)
\end{aligned}$$

where f_{dom} is the true labeling function domain classfication task (outputs 0 for samples come from source domain, otherwise 1), and $\widehat{\mathcal{E}}^{l}_{dom}$ is the corresponding empirical loss. By minimizing $\widehat{\mathcal{E}}^{l}_{dom}$ with respect to h_{dom}, we can obtain a good estimation for $\widehat{\text{disc}}_{\ell_{\mathcal{H}\Delta\mathcal{H}}}$. However, we also need to minimize $\widehat{\text{disc}}_{\ell_{\mathcal{H}\Delta\mathcal{H}}}$ by maximizing $\widehat{\mathcal{E}}^{l}_{dom}$ with respect to $\phi(\mathbf{x})$, thus the min-max optimization problem. The loss function ℓ is the loss of a logistic regressor defined by h_{dom} as:

$$\mathcal{L}_{dann} = -d_i log\left(h_{dom}(\phi(\mathbf{x_i}))\right) - (1-d_i) log\left(1 - h_{dom}(\phi(\mathbf{x_i}))\right), \tag{2.12}$$

where $d_i = 0$ if $\mathbf{x_i} \sim \mathbf{P}_{\mathcal{S}}(\mathbf{X})$ and $d_i = 1$ if $\mathbf{x_i} \sim \mathbf{P}_{\mathcal{T}}(\mathbf{X})$.

To sum up, the training objective of DANN is as follow:

$$\min_{\theta_\phi} \mathcal{L}_{task}(\theta_\phi) - \lambda_d \mathcal{L}_{dann}(\theta_\phi) \tag{2.13}$$

$$\min_{\theta_h} \mathcal{L}_{dann}(\theta_h) \tag{2.14}$$

An intuitive interpretation is that DANN tries to simultaneously learn a classifier that predict where unlabelled data come from and a feature extractor ϕ that not only extracts discriminative features for the main task, but also tries to fool the classifier. The optimal solution is a network that extracts domain-invariant features that the domain critic is unable to classify, but contain useful information for the task at hand.

2.3.2 IPM-based Approaches

While divergence-based approaches have brought about many substantial empirical results, there are certainly some limitations which they cannot overcome because of the theoretical assumptions of the corresponding bound. First of all, the previous methods approximate the divergence between domains based on the considered hypothesis class, without taking into account the geometry of the actual data distributions. Aside conceptual aspect, the computation of $\mathcal{H}\Delta\mathcal{H}$-divergence is an intractable problem, and it is impossible to find the true divergence. Another line of research explores the usage of integral probability metrics (IPMs) as an alternative for quantifying the amount of discrepancy between domains.

Integral probability metrics (IPMs) Given two probability distribution $\mathbf{P}_\mathcal{S}(\mathbf{X})$ and $\mathbf{P}_\mathcal{T}(\mathbf{X})$ defined on the input space $\mathbf{X}$. Let $\mathcal{F}$ be a class of bounded measurable functions $f : \mathbf{X} \to \mathbb{R}$. Then, the IPM between $\mathbf{P}_\mathcal{S}(\mathbf{X})$ and $\mathbf{P}_\mathcal{T}(\mathbf{X})$ defined by $\mathcal{F}$ as follow

$$d_\mathcal{F}\left(\mathbf{P}_\mathcal{S}(\mathbf{X}), \mathbf{P}_\mathcal{T}(\mathbf{X})\right) = \sup_{f \in \mathcal{F}} \left| \int_\mathbf{X} f d\mathbf{P}_\mathcal{S}(\mathbf{X}) - \int_\mathbf{X} f d\mathbf{P}_\mathcal{T}(\mathbf{X}) \right| \tag{2.15}$$

IPMs is a large class of distances defined on the space of probability distributions. Given some specific function class $\mathcal{F}$, the corresponding IPM instance can have properties that are completely different from those of $\mathcal{H}\Delta\mathcal{H}$-divergence, but are useful for computing divergence.

IPM-Bound In the context of domain adaptation, the distance $d_\mathcal{F}\left(\mathbf{P}_\mathcal{S}(\mathbf{X}), \mathbf{P}_\mathcal{T}(\mathbf{X})\right)$ is equal to zero if unlabelled data come from the same distribution $\mathbf{P}_\mathcal{S}(\mathbf{X}) = \mathbf{P}_\mathcal{T}(\mathbf{X})$. The bound of the difference between generalized error of source and target domains was introduced by [Zol83] as follow:

$$\sup_{h \in \mathcal{F}} \left| \mathcal{E}^\ell_{\mathbf{x},\mathbf{y}\sim\mathcal{S}}(h) - \mathcal{E}^\ell_{\mathbf{x},\mathbf{y}\sim\mathcal{T}}(h) \right| \leq d_\mathcal{F}\left(\mathbf{P}_\mathcal{S}(\mathbf{XY}), \mathbf{P}_\mathcal{T}(\mathbf{XY})\right) + \mathbf{C}(\mathcal{F}), \tag{2.16}$$

where the last term of the right-hand side is the complexity term that depends on the covering number of the space $\mathcal{F}$. It can be shown that for a finite complexity term, the difference between the empirical source risk and the target risk converge to zero as number of available source samples increases.

From 2.16, we can see several distinction compared to the divergence-based bounds in 2.2 and 2.20. First, the IPM-Bound is defined for the joint distributions $\mathbf{P}_\mathcal{S}(\mathbf{XY})$ and $\mathbf{P}_\mathcal{T}(\mathbf{XY})$, thus is impossible to estimate in case of UDA. Second, while divergence-based bounds are restricted to 0-1 loss function and a symmetric hypothesis class,

the functional class $\mathcal{F} = \{(\mathbf{x}, y) \to \ell(f(\mathbf{x}), y)\}$ is not explicitly specified here. This leads to the flexibility of the bound, at the same time the need for careful consideration when choosing the appropriate functional class $\mathcal{F}$.

Wasserstein distance Inspire by optimal transportation (OT) theory, [ZZY12] proposed to address above issues by explicitly consider a particular function class based on Wasserstein distance between the marginal distributions for source and target domains. The main goal was to learn the optimal coupling matrix (transport plan) that transports samples from source to target domain with minimal cost defined by the Wasserstein distance:

$$d_{\mathcal{W}}\left(\mathbf{P}_{\mathcal{S}}(\mathbf{X}), \mathbf{P}_{\mathcal{T}}(\mathbf{X})\right) = \min_{\gamma \in \Pi(\mathbf{P}_{\mathcal{S}}(\mathbf{X}), \mathbf{P}_{\mathcal{T}}(\mathbf{X}))} \langle C, \gamma \rangle_F, \tag{2.17}$$

where $\Pi\left(\mathbf{P}_{\mathcal{S}}(\mathbf{X}), \mathbf{P}_{\mathcal{T}}(\mathbf{X})\right) = \left\{\gamma \in \mathbb{R}_{+}^{N_S \times N_T} \mid \gamma \mathbf{1} = \mathbf{P}_{\mathcal{S}}(\mathbf{X}),\ \gamma^T \mathbf{1} = \mathbf{P}_{\mathcal{T}}(\mathbf{X})\right\}$ is a set of doubly stochastic matrices (here the two marginal distributions are discrete empirical measures whose supports are sets of sizes N_S and N_T), and C is a dissimilarity matrix with element $C_{ij} = c(\mathbf{x}_i^S, \mathbf{x}_j^T)$ corresponds to the cost needed to move $\mathbf{x}_i^S$ to $\mathbf{x}_j^T$. This optimization problem admits a unique solution:

$$\gamma^* = \underset{\gamma \in \Pi(\mathbf{P}_{\mathcal{S}}(\mathbf{X}), \mathbf{P}_{\mathcal{T}}(\mathbf{X}))}{\operatorname{argmin}} \langle C, \gamma \rangle_F, \tag{2.18}$$

The usage of Wasserstein distance preserves the topology of data with respect to the Euclidean distance between examples, therefore enabling direct transformation from source samples into target-aligned source samples using the optimal coupling γ^* as:

$$\mathbf{P}_{\mathcal{T}}(\mathbf{X}) = \widehat{\mathbf{P}_{\mathcal{S}}}(\mathbf{X}) = diag\left((\gamma^* \mathbf{1})^{-1}\right) \gamma^* \mathbf{P}_{\mathcal{S}}(\mathbf{X}) \tag{2.19}$$

W_1-Bound Following the same principle as previous bound, one can show that, under certain assumptions and the condition that $\|\ell\| \leq 1$, then the following holds:

$$\forall h \in \mathcal{H},\ \mathcal{E}_{\mathbf{x} \sim \mathcal{S}}^{\ell}(h) \leq \mathcal{E}_{\mathbf{x} \sim \mathcal{S}}^{\ell}(h) + \widehat{d_{\mathcal{W}}}\left(\mathbf{P}_{\mathcal{S}}(\mathbf{X}), \mathbf{P}_{\mathcal{T}}(\mathbf{X})\right) + \lambda_h^* + \mathbf{C}(m), \tag{2.20}$$

where λ_h^* is the combined risk of the ideal hypothesis h^* which minimize $\mathcal{E}_{\mathbf{x} \sim \mathcal{S}}^{\ell}(h) + \mathcal{E}_{\mathbf{x} \sim \mathcal{T}}^{\ell}(h)$, and $\mathbf{C}$ a is function independent of h and goes to zero as number of sample m increases.

Wasserstein Distance Guided Representation Learning (WDGRL) The work of [She+18] leveraged the usefulness of Wasserstein distance in neural network methods for unsupervised domain adaptation. WDGRL followed the same framework as DANN. However, instead of domain-adversarial training, [She+18] proposed a mechanism to estimate the Wasserstein-1 distance between the marginal distributions of source and target domains. In particular, the Wasserstein-1 distance written as a form of IPM as

$$d_{\mathcal{W}}\left(\mathbf{P}_{\mathcal{S}}(\mathbf{X}), \mathbf{P}_{\mathcal{T}}(\mathbf{X})\right) = \sup_{\|f\|_L \leq 1} \mathbf{E}_{\mathbf{P}_{\mathcal{S}}(\mathbf{X})}\left[f(\mathbf{x}^{\mathcal{S}})\right] - \mathbf{E}_{\mathbf{P}_{\mathcal{T}}(\mathbf{X})}\left[f(\mathbf{x}^{\mathcal{T}})\right] \tag{2.21}$$

where the Lipschitz semi-norm is defined as $\|f\|_L = \sup |f(x) - f(y)| / \rho(x, y)$. Using a domain critic $f_w : \mathbb{R}^d \to \mathbb{R}$ that maps the feature representations h, computed

by feature extractor $h = f_g(x)$, The distance in 2.21 for two representation distribution $\mathbb{P}_h s$ and $\mathbb{P}_{h^t}$, where $h^s = f_g\left(x^s\right)$ and $h^t = f_g\left(x^t\right)$, formulated as:

$$d_{\mathcal{W}}\left(\mathbb{P}_{h^s}, \mathbb{P}_{h^t}\right) = \sup_{\|f_w\|_L \leq 1} \mathbb{E}_{\mathbb{P}_h^s}\left[f_w(h)\right] - \mathbb{E}_{\mathbb{P}_h^t}\left[f_w(h)\right] \tag{2.22}$$

$$= \sup_{\|f_w\|_L \leq 1} \mathbb{E}_{\mathbf{P}_{\mathcal{S}}(\mathbf{X})}\left[f_w\left(f_g(\mathbf{x}^{\mathcal{S}})\right)\right] - \mathbb{E}_{\mathbf{P}_{\mathcal{T}}(\mathbf{X})}\left[f_w\left(f_g(\mathbf{x}^{\mathcal{T}})\right)\right]. \tag{2.23}$$

This distance can be approximated empirically if the parameterized family of domain critic function $\{f_w\}$ are all 1-Lipschitz, through the following domain critic loss:

$$\mathcal{L}_{wd}\left(x^s, x^t\right) = \frac{1}{n^s} \sum_{\mathbf{x} \sim \mathcal{S}} f_w\left(f_g\left(x^s\right)\right) - \frac{1}{n^t} \sum_{\mathbf{x} \sim \mathcal{T}} f_w\left(f_g\left(x^t\right)\right) \tag{2.24}$$

Replacing the optimization process of domain classifier in DANN with the domain critic counterpart, the model can iteratively learn to lower Wasserstein distance and making the shared features discriminative to the main task, thus effectively producing domain-invariant feature representations.

Chapter 3

Domain Adaptation In Natural Language Processing

Linguistic data can vary in many dimension, from symbolic (languages) to syntactic (structures) and semantic (meanings). Thus, building a general model that is able to adapt to these change is one of the central goals of NLP research. This chapter starts with an overview on current state-of-the-art language models (LM) and attention architecture (§ 3.1). In particular, we review the architecture of the most popular variant of self-attention based LMs - Bidirectional Encoder Representations from Transformers (BERT). Then, we study the transferability of BERT's representations from different perspectives of the two training procedures - pretraining and finetuning (§ 3.2). Finally, domain alignment approaches for text classification task are introduced, together with several analyses that motivated our proposed solution for the corresponding domain adaptation task (§ 3.2).

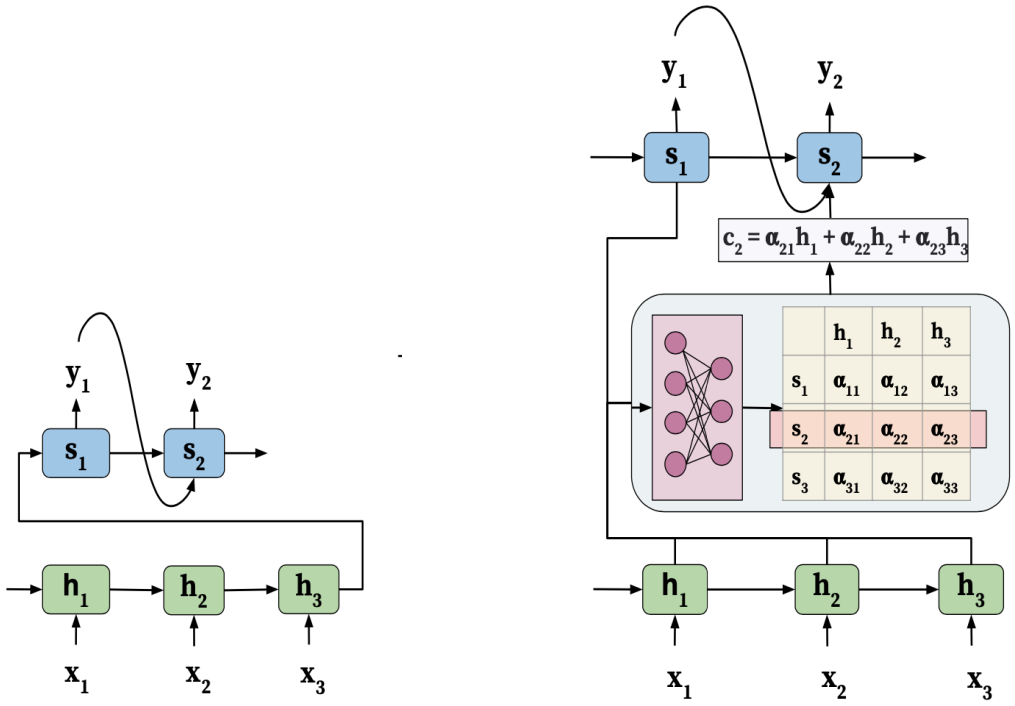
3.1 Large Language Models

3.1.1 Self-Attention and Transformer Architecture

Attention mechanism for neural network was first introduction in the context of Machine Translation by [BCB14]. From then on, attention has become one of the main ingredients for the success of remarkably large number of applications in Computer Vision, Speech Recognition, and especially NLP. Inspire by the way human efficiently process information by focusing on the most relevant signals while ignoring other noises which are plethora in real world inputs, attention mechanism main goal is to help models ranking the importance of different parts of inputs, thus enabling selective training using the only the ones that are most beneficial to the task at hand.

Earlier works that leveraged attention mechanism mostly focus on cross-attention in the context of auto-encoder architecture, where decoder's outputs are represented as weighted sum of encoder's inputs. The weights here can be interpreted as the amount of attention of the current output and are proportional to the similarity between it and each of the encoder's inputs (see figure 3.1 for example).

The advantages of using attention mechanism are multi-fold. Aside from better extraction of relevance information, it is also bring interpretability into the black-box deep neural networks, of which remarkable performance is impossible to understand. However, the most notable contribution of the mechanism is the replacement of Recurrent Neural Networks(RNNs) with Multi-head Self-Attention Layers as the main components for the task of language modelling. Attention-based LM not only overcomes the un-parallelable sequential nature of RNN, but also addresses the issue of modeling context of extremely long sentences, dependency of which the RNN models are unable to learn due to optimization problems. Thus, the computational



(A) Naive Auto-Encoder

(B) Auto-Encoder with Attention

FIGURE 3.1: Attention in Auto-Encoder (from [Cha+19]) . Figure (a) is the naive setting of an auto-encoder. Figure (b) is an auto-encoder with attention mechanism.

Function	Traditional Encoder-Decoder	Encoder-Decoder with Attention
Keys	$h_i = f\left(x_i, h_{i-1}\right)$	$h_i = f\left(x_i, h_{i-1}\right)$
Values	$c = h_T$	$c_j = \sum_{i=1}^{T} \alpha_{ij} h_i$
Weights		$\alpha_{ij} = p\left(e_{ij}\right)$
Attentions		$e_{ij} = a\left(s_{j-1}, h_i\right)$
Queries	$s_j = f\left(s_{j-1}, y_{j-1}, c\right)$	$s_j = f\left(s_{j-1}, y_{j-1}, c_j\right)$
Generate	$y_i = g\left(y_{i-1}, s_j, c\right)$	$y_j = g\left(y_{j-1}, s_j, c_j\right)$

$x = (x_1, \ldots, x_T)$: input sequence ; T : length of input sequence ; h_i : hidden states of encoder ; c : context vector, α_{ij} : attention weights over input ; s_j : decoder hidden state ; y_j : output token ; f, g : non-linear functions ; a : alignment function ; p : distribution function.

TABLE 3.1: Attention Architectures

efficiency and modeling capability of attention-based LM are both superior compare to the its counterpart.

Self-Attention Self-attention, also referred to as intra-attention, is an attention mechanism where a sequence focuses on itself, building a contextual representation for each of its position as a weighted sum of the others. Though successfully combined with RNN-based language models for a variety of tasks, its most notable contribution is when [Vas+17] introduced Transformer, a language model completely relied on attention mechanism. By doing away with the recurrence inductive bias of previous approaches, Transformer managed to achieve significantly more parallelization and trainability, therefore enabling the model to have substantially more capacity and to be trained effectively on large corpora. Formally, Transformer used a Scaled Dot-Product Attention mechanism, which is a normalized matrix version of the original attention:

$$\text{Attention}\left(Q, K, V\right) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{3.1}$$

where Q and K are matrix of queries and keys of dimension d_k, and V is value matrix of vector with dimension d_v. Because of this formulation, it is possible to achieve time and space-efficient learning using highly optimized matrix multiplication code.

Multi-Head Attention To enhance the representation capability of self-attention mechanism, Transformer jointly attended to information from different feature subspaces at different positions by creating multiple linear projections of Q, K and V into a portion of their initial dimensions

$$\text{MultiHead}(Q, K, V) = \text{Concat}\left(\text{head}_1, \ldots, \text{head}_\text{h}\right) W^O \tag{3.2}$$

$$\text{head}_\text{j} = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right), \tag{3.3}$$

where $W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$ are weights of the linear layers. In 3.2, attention on each of the projected version is performed in parallel, while the total computation cost remained the same due to the reduced dimension of each head. As a result, multi-head attention can simultaneously improve performance and efficiency.

3.1.2 Bidirectional Encoder Representations from Transformers (BERT)

Previous directional language models were able to leverage inductive bias of RNN architectures to learns meaningful words' representations. However, it also severely restrict the power of the pretrained self-attention based representations, especially for the finetuning approaches, because every token can only attended to previous tokens. Bidirectional Encoder Representations from Transformers (BERT) was proposed in [Dev+18] to address this constraint. They proposed a new pretraining objective - Masked Language Model (MLM) - in which the model randomly masks some of the tokens from the input, and the goal is to predict the original vocabulary id of the masked word based only on its context. MLM does away with the recurrence, allowing representations to be learned at arbitrary positions of the context, resulting in a deep bidirectional model.

BERT's Architecture Fundamentally, BERT is a stack of multi-head self-attention based encoders base on Transformer. Each layer contains two primary sub-layers:

an self-attention layer and a feed-forward layer. Another fully connected layer is added on top of individual sub-layers to project the features size back to the hidden dimension of inputs' contexts. Furthermore, skip-connection is applied across each of the sub-layers and their final outputs is are fed into layer normalization. There are two original version of BERT: base and larger, varying in the number of layers, their hidden size, and number of attention heads.

BERT's Learning Procedure Typically, a BERT model is trained through two stages: pretraining and finetuning. Pretrainings use unsupervised task of MLM on extremely large unlabelled corpora to make BERT's representation contextual. On the other hand, when finetuning BERT for smaller downstream tasks with fully annotated datasets, one or more feed-forward layers are added on top of the final encoder layer to adapt the general knowledge of a pretrained BERT into more task-specific features.

While BERT and self-attention based LM in general are widely adopted in current NLP field, full understanding of their contextual representation is still an ongoing research. BERTology, which refers to studies that focus on investigating the various aspects of BERT models, have resulted in many interesting facts surrounding the types of knowledge learned by BERT, where this knowledge is represented, how it is learned, and the methods proposed to improve it. We summarize the findings related to layers of BERT here and refer to [RKR21] for a comprehensive review of BERTology to date.

Lower Layers - Statistical Information The first few layers of BERT receives as input a combination of token, segment, and positional embeddings. Thus, lower layers contain mostly general statistical knowledge such as linear word order information. This information decreases for going up in layers accompanied by increased knowledge of more sophisticated hierarchical sentence structure.

Middle Layers - Syntactic Knowledge The middle layers of BERT are the most transferable and overall best-performing across tasks. Embedding of these layers contain hierarchical information about parts of speech, syntactic chunks and roles, i.e. similar (but not exactly the same) to a syntactic tree structure. As a result, BERT's encoding of syntactic structure can be transformed to represent the actual linguistic counterpart.

Final Layers - Semantic Knowledge The final layers of BERT are the closest to the learning of the main task, hence the most task-specific. In particular, they encodes semantic knowledge such as entity types, relations, semantic roles, and proto-roles. Because these layers contain most of information regarding the task at hand, they changed the most in finetuning process and are responsible for most of the downstream performance. In pretraining, this also means specificity to the MLM task, which would explain why the middle layers are more transferable.

BERT's Embedding A token's learned embedding refers to the corresponding output vector of BERT's final layer. These embedding are contextualized, in the sense that each of them is represented by a vector dependent on the particular context of its occurrence. Studies have shown the basic distributional hypothesis, similar to

that of statistical learned embeddings, is also applied for BERT's contextualized embeddings. This means BERT's representations formed distinct and clear clusters corresponding to word senses. Furthermore, they occupy a narrow cone in the vector space, hence two random words will on average have a much higher cosine similarity than expected, and this effect increases from lower to higher layers (moving from general to context-specific representations).

Over-parametrization Since the time Transformer was introduced, attention-based models' capabilities have improved substantially, yet at the cost of significant increase in size, where the current state-of-the-art model has over 30 times the number of parameters of BERT-base. While an argument is made about the extreme complexity of language in general, multiple works have shown that it is possible to reduce the capacity of BERT without serious impact on overall performance. They provided observations that most heads in the same layer show similar self-attention patterns, and some are even useless or harmful to downstream task. Another line of researches was able to compress BERT models into a third of their original sizes without significant losses in performance.

3.2 Transfer Learning with BERT

3.2.1 Domain-Related Pretraining

BERT not only excels at capturing context using unsupervised tasks, but also is able to transfer the self-learned knowledge to downstream tasks, completely outperforms previous approaches by a wide margin in multiple benchmarks of natural language understanding. Because of BERT pretraining ability, several natural questions arise: How to best leverage unlabelled data to improve a particular task? What would be the requirements of the unsupervised dataset so that the pretraining procedure will have a positive effect? Is the more data, the better a pretrained model performs? Or are there certain aspects of a corpus that have more profound effects compare to others? To investigate these questions, [Gur+20] conducted various experiments on pretraining BERT using different settings and provided useful insights how the benefit of continued pretraining may vary with factors like the amount of available labeled task data, or the proximity of the target domain to the original pretraining corpus (see Figure 1).

Domain-adaptive pretraining (DAPT) The idea of domain-adaptive pretraining (DAPT) is straightforward: continue pretraining BERT LMs on a large corpus of unlabelled domain-specific text.

Task-adaptive pretraining (TAPT) When building a dataset with a specific tasks in mind, the resulting corpus tend only cover a subset of the available text that satisfied annotators' requirements. To answering the question of whether it is helpful to pretrain model on task-relevant unlabelled data, task-adaptive pretraining is formulated. TAPT presents a constrasting trade-off with DAPT: lower quantity (smaller corpus) - higher quality (more closely related to the concerning task). As a result, TAPT is much more computation efficient to run than DAPT. Furthermore, experiments conducted following TAPT procedure provided results that is approximately equal, some time even exceed those of DAPT.

Finally, it is also possible to combine DAPT and TAPT together, first conditioning model on domain-related corpus, then finetuning model on task-related data. Though this sequential approach outperformed both of its singular counterparts, the improvement was too negligible taking into account the computation cost it required.

3.2.2 Adapter-based Finetuning

Another approach to adapt a pretrained LM to a new task is to finetune on that task's dataset. Whilst this method can improve the model's performance on the task at hand, it can also completely bias the LM to create representations that are overfitted to the trained task because of the high capacity of the whole architecture. For it to be beneficial to transfer these learned knowledge to another task, the entire model needs to be retrained for that specific task, which is not only time-inefficient, but also impractical in case where there are many tasks or when we are learning in a streaming setting. Even more damaging is when we are trying to transfer between two highly incompatible tasks, in which case optimizing for one task only reduces the generality of the pretrained LM and the model has to unlearn the previous task's features to transfer to the other task. Thus, [Hou+19] proposed Adapter-based finetuning (AbF), which is an alternative to fully finetuning, to address all of the above issues.

Adapter-based finetuning [Hou+19] suggest three key properties for an effective an tuning strategy of a large LM on multiple downstream tasks

1. Performance The finetuned model's extracted features should be general and able to transfer to target downstream task.

2. Sequentiality The training procedure should not require simultaneous access to all datasets, thus enabling stream learning setting.

3. Efficiency The increment of parameters per task should be negligible with respect to the overall architecture.

To address all the above concerns, AbF firstly loads and fixes all LM's parameters that was trained previously in unsupervised manner. Formally, consider a pretrained BERT model in particular, which composed of N similar blocks of layers stacked hierarchically $b_i(\mathbf{x})$ for $i \in \mathbb{N}$, $0 \leq i \leq N-1$. A naive finetuning process consists of add a new target classifier h_{target} on top, producing the final output $h_{target}(b_{N-1} \circ \cdots \circ b_0(\mathbf{x}))$ which is used by loss function. Learning involves both optimizing parameters of h_{target} and adjusting the original parameters of b_i, which account for nearly 99% of the tunables. In contrast, AbF manages to sidestep the need to totally relearn the unwieldy BERT by injecting new modules called adapters $a_i(\mathbf{x})$ for $i \in \mathbb{N}, 0 \leq i \leq N-1$, into each of the corresponding b_i. These adapters take on the task of adapting b_i layers, which are now fixed, to the target task. The concept of layer-wise adaptations is very much general where ones can modify the original layers in various ways to achieve some desired properties. For simplicity, we only consider a_i as a small neural network taking the representation computed by b_i as input. The overall augmented architecture is a layer-wise refinement of the original LM, while also keeping the hierarchical structure on which every deep learning models depends for their remarkable representation power. If the adapters are designed so that their total number of parameters $\mathbf{N}_a \ll \mathbf{N}_b$, the size of BERT, then the

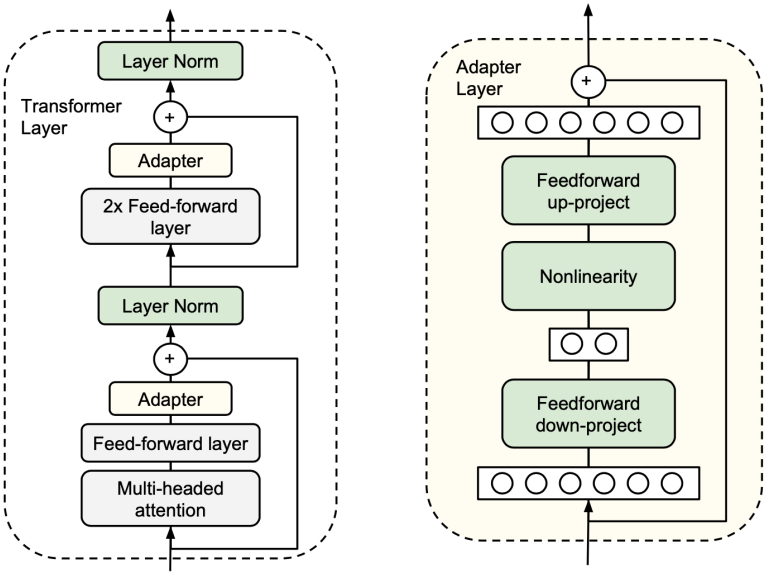


FIGURE 3.2: Adapter Architecture for BERT's layer (from [Hou+19]

effective adaptation model would requires $\sim \mathbf{N}_b$ for considerable amount of tasks. Furthermore, as b_i remains unchanged and new a_i is added, AbF can sequentially learn new tasks without forgetting the preceding ones. In consequence, in order for AbF to work properly, designing adapter layers to satisfy above conditions is of the utmost importance.

Adapter for BERT's layer Due to the composite properties of BERT's architecture, there are various ways to augment BERT's individual layer with adapter. The most intuitive approach would be to inserting adapters after each of the layer's sub-layer, creating $a_i(b_i^{s-attn}(\mathbf{x}))$ and $a_i(b_i^{fcn}(\mathbf{x}))$. For consistency, the adapter always preserves the context hidden dimension by taking to the output of the sub-layer, after the projection back to the input size, as input. Then, skip-connection and layer normalization follow as in the original model. To limit the number of additional parameters, [Hou+19] propose a bottleneck architecture, which consists of down-sampling projection layer (to smaller dimension c) up-sampling projection layer (back to same dimension d as context) - $a_i = a_{up} \circ a_{down}$. By setting $m << d$, the amount of added parameters per layer, which equals $2md + d + m$ (including biases), are only a fraction of the size of BERT's layer. The bottleneck dimension c defines a capacity vs efficiency trade-off, and choosing c from 5% to 10% of d can give satisfying result for both sides. Finally, to preserve the performance of original model in case adapters are not need, near-zero initialization scheme is used with additional skip-connection around the added modules.

Effects of adapter-based finetuning [Hou+19] provides several notable observations regarding adapter's architecture and capability. First of all, consider a AbF trained model, individual adapter layer have yield little to none impact on the overall model. However, the more adapters are removed, the worse the augmented model performs. In addition, most of the improvement comes from adapters of higher-level layers, which is consistent with naive finetuning setting where more

focus is placed on near-classifier layers that contain most of the task-specific knowledge. This confirms the fact to adapter modules are also able to adjust the amount transfer needed layer-wise, where lower-level layers extracting general features require rarely any adaptation compare to lower-level ones. Last but not least, varying the adapter sizes have small effect on the quality of the model with respect to any given tasks. Thus, a fixed c could be used across different tasks with little damage to model's performance.

3.3 Domain Adaptation Approaches for Text Classification

In the field of computer vision (CV), images from different domain exhibits many inner traits which are easy to identify even by normal human, thus making the task of domain adaption simple and straightforward to evaluate. In contrast,the term domain in NLP can refer to one of many different aspects of language such as genre (e.g. news, novels, academic publications), style (e.g. formal, informal, dialogue, transaction), origin (e.g. text collected from a particular source, or even from a specific demographic). Each of them may affect the context of corpus in diverse ways, which in turn shift the marginal distribution, or even the joint distribution differently. As a result, analyzing performance of a domain adaption algorithm for NLP can be very challenging.

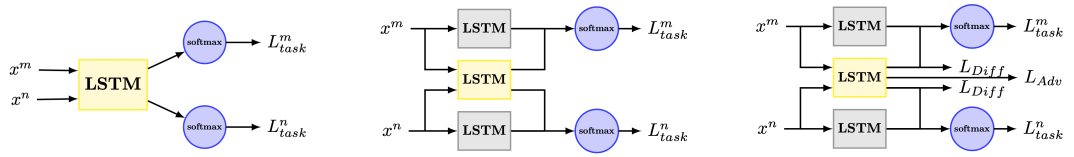
3.3.1 Domain-Adversarial Neural Networks (DANN)

while DANN (2.3.1) and its variants are very well-studied approaches in CV's domain adaption researches, theirs NLP counterparts are pale in comparison, especially for newly established architectures like BERT. There have been only several works that adopt DANN to align the contextualized representations learned by BERT across domains ([NR20], [WA20]), which report little to none improvement on target capability of BERT. One of the main reasons is that the contextual representations of text is highly dimensional. As a result, a naive implementation of domain-adversarial training for NLP tasks will encounter most of the mentioned issues of DANN (2.3.1), and unable to produce domain-invariant features, or even fail to converge.

3.3.2 Domain Separation Network (DSN)

Motivated by theory on shared–space component analysis and the fact that a model that is able to extract domain-independent feature should also has the ability to express information that is unique to each domain, Domain Separation Network ([Bou+16], [LQH17]) tries to explicitly separate feature extractor into a shard encoder ($\mathbf{E}_c$) and different private encoders ($\mathbf{E}_p^{\mathcal{S}}$ and $\mathbf{E}_p^{\mathcal{T}}$). To achieve the desired factorization on the space of input representations, DSN minimizes the divergence of the marginal distributions computed from the shared encoder $\mathbf{E}_c\left(\mathbf{x}^{\mathcal{S}}\right)$ and $\mathbf{E}_c\left(\mathbf{x}^{\mathcal{T}}\right)$ while also maximally disentangling the domain-specific representations $\mathbf{E}_p^{\mathcal{S}}\left(\mathbf{x}^{\mathcal{S}}\right)$ and $\mathbf{E}_p^{\mathcal{T}}\left(\mathbf{x}^{\mathcal{T}}\right)$ from their $\mathbf{E}_c$ counterparts. The former optimization problem uses a DANN as described above

$$\mathcal{L}_{similarity} = \mathbf{DANN}\left(\mathbf{E}_c\left(\mathbf{x}^{\mathcal{S}}\right), \mathbf{E}_c\left(\mathbf{x}^{\mathcal{T}}\right)\right), \tag{3.4}$$



(A) Fully Shared Model (B) Shared-Private Model (C) DSN model

FIGURE 3.3: Multitask NLP Settings (from [LQH17]) . Figure (a) is the naive setting where a single architecture responsible for all task. Figure (b) is the shared-private setting where the task-specific representations are explicitly separated from the shared representations. Figure (c) is the DSN architecture where the shared encoder leverage a DANN to learn task-invariant features.

while the latter can also be formulated as a minimization of Frobenius norm of the multiplication between shared representations and others representations

$$\mathcal{L}_{difference} = \left\| \mathbf{E}_{\mathcal{C}} \left(\mathbf{x}^{\mathcal{S}} \right)^{\top} \mathbf{E}_{p}^{\mathcal{S}} \left(\mathbf{x}^{\mathcal{S}} \right) \right\|_F^2 + \left\| \mathbf{E}_{\mathcal{C}} \left(\mathbf{x}^{\mathcal{T}} \right)^{\top} \mathbf{E}_{p}^{\mathcal{T}} \left(\mathbf{x}^{\mathcal{T}} \right) \right\|_F^2 . \tag{3.5}$$

Equation 3.5 can be interprets as a soft subspace orthogonality constraint. However, while $Enjx\mathbf{x}^{\mathcal{S}}$ are trained to be discriminative for the main task, the others domains' representations are basically task-free and, which minimization of $\mathcal{L}_{difference}$ could lead to a trivial solution where both are simple pushed to the complementary space of $\mathbf{E}_{\mathcal{C}}(\mathbf{x})$ while containing no useful knowledge. To address this issue, DSN adds constraint to the private encoders by using their recombining with the corresponding joint representations to learn a reconstruction task in unsupervised regression manner

$$\mathcal{L}_{recon} = \mathcal{L}_{reg} \left(\mathbf{E}_{\mathcal{C}} \left(\mathbf{x}^{\mathcal{S}} \right) + \mathbf{E}_{p}^{\mathcal{S}} \left(\mathbf{x}^{\mathcal{S}} \right) \right) + \mathcal{L}_{reg} \left(\mathbf{E}_{\mathcal{C}} \left(\mathbf{x}^{\mathcal{T}} \right) + \mathbf{E}_{p}^{\mathcal{T}} \left(\mathbf{x}^{\mathcal{T}} \right) \right) . \tag{3.6}$$

The final resulting representations are $\mathbf{E}_{p}^{\mathcal{S}}$ and $\mathbf{E}_{p}^{\mathcal{T}}$ with contain domain-specific information to rebuilt the original input, while the $\mathbf{E}_{\mathcal{C}}$ are also empowered being part of the reconstructions from both domains. The overall objective of the training procedure is

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_r \mathcal{L}_{\text{recon}} + \lambda_d \mathcal{L}_{\text{difference}} + \lambda_s \mathcal{L}_{\text{similarity}} , \tag{3.7}$$

where $\lambda_r, \lambda_d, \lambda_s$ are hyper-parameters that control the interaction of the loss terms.

Shared Representations of Multiple Tasks The work of [LQH17] studied the transferrability of shared representations learned from DSN for several NLP tasks in the context of multitask setting. While in most naive approaches, the common features are usually contaminated by task-specific signals or the noise brought by the diversity of all the tasks' datasets. [LQH17] showed that despite having to adapt to numerous distinct tasks, DSN still managed to retain a good task-independent representation that are not only general to all current tasks and but also useful for learning new tasks.

Remark One apparent weakness of DSN in context of domain adaptation is that task classifier only has access to the domain-invariant features only, while the target representations may contain beneficial information to improve performance on target domain.

Chapter 4

Proposed Method: Adapter-based Hierarchical Domain Separation

We present our proposed approach for text classification task in unsupervised domain adaptation setting - Adapter-based Hierarchical Domain Separation (AHDS) - and its detail implementation in this chapter. Overall, AHDS consists of 5 main modules: Adapter-based Domain-specific Encoders $\mathbf{A}^{\mathcal{S}}, \mathbf{A}^{\mathcal{T}}, \mathbf{A}^{\mathcal{J}}$ (§ 4.1), Data Selection Head ($h^{\mathcal{ST}}_{wass}$) (§ 4.2), Domain Critic Head $h^{\mathcal{J}}_{dann}$(§ 4.3.1), Domain Separation Heads $h^{\mathcal{STJ}}_{sim}$ and $h^{\mathcal{STJ}}_{mlm}$ (§ 4.3.2), and Task Classification Head $h^{\mathcal{J}}_{task}$ (§ 4.4). We utilize the AbF approach (§ 3.2.2) to train the model for the task of domain adaptation for text classification, which proceeds in these following steps:

- **Initialization**: The overall problem can be formulated as: Given a source dataset of tuples $\mathcal{D}^{\mathcal{S}}_{\mathbf{XY}} = \{(\mathbf{x}, \mathbf{m}, \mathbf{y})\}_{\mathbf{x},\mathbf{y}\sim\mathbf{P}_{\mathcal{S}}(\mathbf{XY})}$ and a target dataset of unlabelled pairs $\mathcal{D}^{\mathcal{T}}_{\mathbf{X}} = \{(\mathbf{x}, \mathbf{m})\}_{\mathbf{x}\sim\mathbf{P}_{\mathcal{T}}(\mathbf{X})}$, where $\mathbf{x}$ is a sequence of words $(x_0, \cdots, x_{N-1})$ of fixed length N, $\mathbf{m}$ is the position of the corresponding trigger word that needed to be classified ($\mathbf{m} \in \mathbb{N}, 0 \leq \mathbf{m} \leq N-1$), and $\mathbf{y}$ is the true label for that word ($\mathbf{y} \in \mathbb{N}, 0 \leq \mathbf{m} \leq K-1$, K is number of possible classes). We use the pretrained BERT's tokenizer and model as the fixed base of our AHDS model. Before training starts, while the label $\mathbf{y}$ is converted into its matching one-hot vector, the input sequence $\mathbf{x}$ goes through the tokenizer and becomes a sequence of bpe tokens (refer to § 3.1 for more details). In this step, a trigger word may be splitted into multiple tokens, thus we make use of only the corresponding first token as our main representation for the downstream task because the self-attention mechanism enables effective gathering of local information in sequence (for brevity, when talking about an input representation from here on, we will only refer to representation of this specific token, unless stating otherwise). We finish the initialization of our AHDS model by adding adapters and heads as listed above with appropriate randomly-created weights $\theta_{\mathbf{A}^{\mathcal{S}}}$, $\theta_{\mathbf{A}^{\mathcal{T}}}$, $\theta_{\mathbf{A}^{\mathcal{J}}}$, θ_{wass}, θ_{dann}, θ_{mlm}, θ_{task}.

- **Tokens Representation**: In the first step of AbF training, a minibatch of samples composed of $n_{\mathcal{S}}$ instances from $\mathcal{D}^{\mathcal{S}}_{\mathbf{XY}}$ and $n_{\mathcal{T}}$ instances from $\mathcal{D}^{\mathcal{T}}_{\mathbf{X}}$ is sampled. Its token sequences are taken by the adapters as inputs depending their domain. Specifically, source adapter $\mathbf{A}^{\mathcal{S}}$ uses $\mathbf{x} \sim \mathcal{S}$, target adapter $\mathbf{A}^{\mathcal{T}}$ uses $\mathbf{x} \sim \mathcal{T}$, while joint adapter $\mathbf{A}^{\mathcal{J}}$ takes as input sequences from both domains (for convenience, we refer to the union of unlabelled data of the two domains as another domain called joint). These adapters produce contextual representations $\mathbf{A}^{\mathcal{S}}\left(\mathbf{x}^{\mathcal{S}}\right)$, $\mathbf{A}^{\mathcal{T}}\left(\mathbf{x}^{\mathcal{T}}\right)$, $\mathbf{A}^{\mathcal{J}}\left(\mathbf{x}^{\mathcal{J}}\right)$ which will be used by the heads.

- **Data Selections**: $h^{\mathcal{ST}}_{wass}$ is a simple neural network that try to estimate the Wasserstein-1 distance between source representations $\mathbf{A}^{\mathcal{S}}\left(\mathbf{x}^{\mathcal{S}}\right)$ and target representations $\mathbf{A}^{\mathcal{T}}\left(\mathbf{x}^{\mathcal{T}}\right)$. Based on this value, we select a subset of $\widehat{n_{\mathcal{S}}}$ source samples with representations as close to the target counterparts as possible.
- **Domain-Adversarial Learning**: The union of the chosen source samples and target samples goes through a modified DANN $h^{\mathcal{J}}_{dann}$ (§ 2.3.1) which aligns the distributions of every layers' representations and outputs domain-invariant features.
- **Domain Separation**: Concurrently, the domain separation mechanism, which consists of a similarity head $h^{\mathcal{ST}\mathcal{J}}_{sim}$ and a masked LM head $h^{\mathcal{ST}\mathcal{J}}_{mlm}$, tries to disentangle the representations of the three adapters so that $\mathbf{A}^{\mathcal{S}}$ and $\mathbf{A}^{\mathcal{T}}$ only encode domain-specific knowledge respectively to their own domains while $\mathbf{A}^{\mathcal{J}}$ gathers information that is independent to its inputs' origin.
- **Source Task Learning**: Finally, task classifier $h^{\mathcal{J}}_{task}$ labels samples from source domain using the domain-invariant representations $\mathbf{A}^{\mathcal{J}}\left(\mathbf{x}^{\mathcal{S}}\right)$ and compare the result with the corresponding true labels provided by $\mathcal{D}^{\mathcal{S}}_{\mathbf{XY}}$.

4.1 Efficient Transfer Learning with Adapter Architecture

4.1.1 Shared Pretrained-BERT

For the base of the AbF training, we make use of the pretrained bert-base model, which was previously optimized for the tasks of masked language model and next sentence prediction in unsupervised manner on several extremely large corpora. The diversity of these unlabelled text also pushes the learned LM to be a good starting point for learning domain-invariant features, which would be lost if we follow the naive finetuning approach.

4.1.2 Domain-specific Adapters

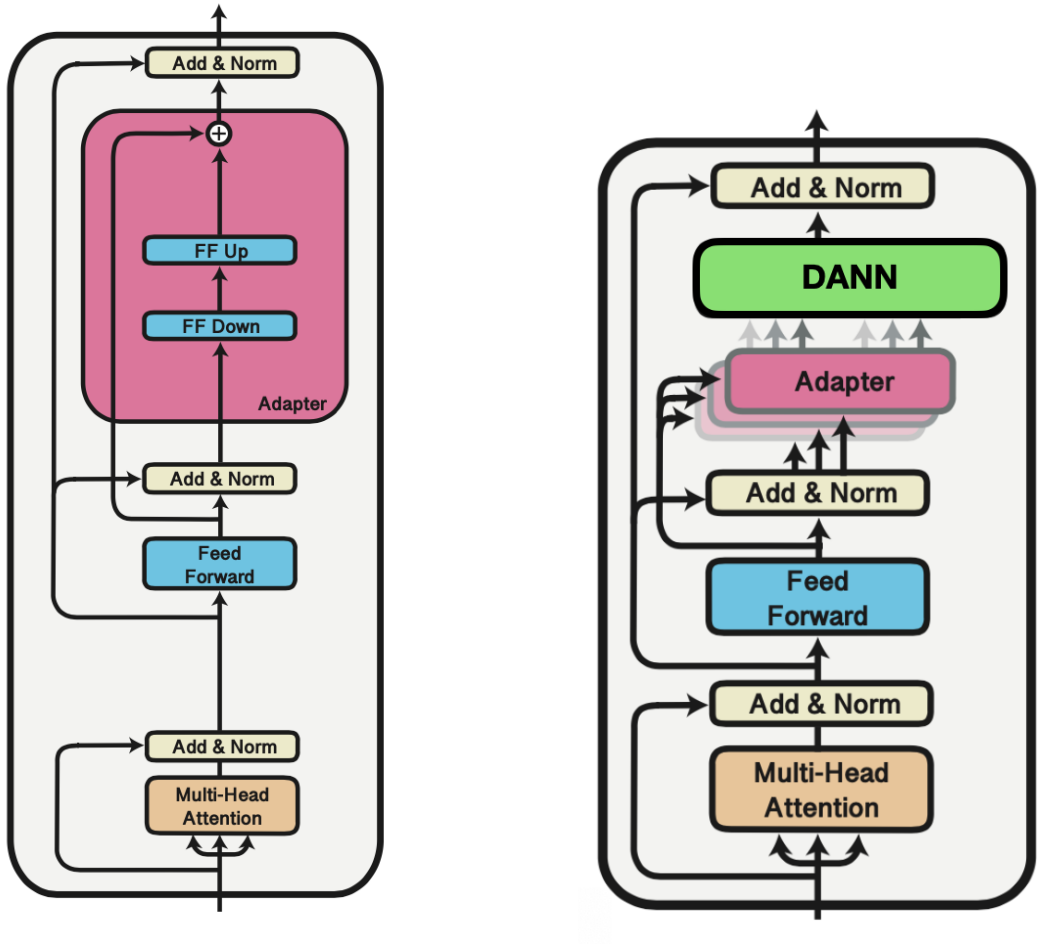
An adapter for each domain: To explicitly create representations for each of the domain, we create three adapter models augmented on the same BERT layers. Formally, we add on top of each BERT's layer b_i three adapter modules $a^{\mathcal{S}}_i$, $a^{\mathcal{T}}_i$, $a^{\mathcal{J}}_i$ with the same architecture, resulting in the overall models $\mathbf{A}^{\mathcal{S}}$, $\mathbf{A}^{\mathcal{T}}$, $\mathbf{A}^{\mathcal{J}}$ that output representations for source data $\mathbf{x}^{\mathcal{S}}$, target data $\mathbf{x}^{\mathcal{T}}$, and joint data $\mathbf{x}^{\mathcal{J}} = \mathbf{x}^{\mathcal{S}} \cup \mathbf{x}^{\mathcal{T}}$ respectively. The joint features $\mathbf{A}^{\mathcal{J}}$ is our main representation which will be used for classification task. On the other hand, the domain-specific representations $\mathbf{A}^{\mathcal{S}}$ and $\mathbf{A}^{\mathcal{J}}$ will only be used to help $\mathbf{A}^{\mathcal{J}}$ find to the optimal joint-domain space while simultaneously retain good performance on source domain. For instance, using BERT-base model:

$$\mathbf{A}^{\mathcal{S}}\left(\mathbf{x}^{\mathcal{S}}\right) = a^{\mathcal{S}}_{11} \circ b_{11} \circ \cdots \circ a^{\mathcal{S}}_{0} \circ b_0(\mathbf{x}^{\mathcal{S}}), \tag{4.1}$$

$$\mathbf{A}^{\mathcal{T}}\left(\mathbf{x}^{\mathcal{T}}\right) = a^{\mathcal{T}}_{11} \circ b_{11} \circ \cdots \circ a^{\mathcal{T}}_{0} \circ b_0(\mathbf{x}^{\mathcal{T}}), \tag{4.2}$$

$$\mathbf{A}^{\mathcal{J}}\left(\mathbf{x}^{\mathcal{J}}\right) = a^{\mathcal{J}}_{11} \circ b_{11} \circ \cdots \circ a^{\mathcal{J}}_{0} \circ b_0(\mathbf{x}^{\mathcal{J}}), \tag{4.3}$$

$$\mathcal{L}^{\mathcal{J}}_{task}\left(\theta_{\mathbf{A}^{\mathcal{J}}}, \theta_{task}\right) = \text{CrossEntropyLoss}\left(h^{\mathcal{J}}_{task}\left(\mathbf{A}^{\mathcal{J}}\left(\mathbf{x}^{\mathcal{S}}\right)\right), f^{\mathcal{S}}\right). \tag{4.4}$$



(A) A Single Bottle-neck Adapter (B) Domain-Specific Adapters

FIGURE 4.1: Domain-Specific Adapters (from [Pfe+20]). In figure (a) shows a single adapter module inside a layer of BERT with bottle architecture. Figure (b) present a multiple adapters setting where each adapter corresponds to a specific domain's representation.

Adapter module's architecture: There are a variety of ways that one can design adapter modules' architecture, following the observations from [Pfe+20], we choose ours to be the most efficient but also effective, which is a singular bottleneck neural network with skip-connection, taking features computed by BERT's feed-forward sub-layer as input. The adapter architecture can be decoupled into two part:

$$a_i = a_i^{up} \circ a_i^{down}, \tag{4.5}$$

$$where \;\; a_i^{down} : \mathbb{R}^d \rightarrow \mathbb{R}^c \;,\; a_i^{up} : \mathbb{R}^c \rightarrow \mathbb{R}^d \;,\; c \ll d. \tag{4.6}$$

As a result, despite tripling the added parameters from adapter modules, the amount that needed to be tuned is still only less than 10% the size of the pretrained BERT model.

4.2 Wasserstein Distance Guided Samples Selection to Alleviate Negative Adaptation

The closer the marginal distributions of source and target domain are, the easiest it is for DANN to align them. Motivated by this fact, we propose to minimize Wasserstein distance between marginal representation distributions $\mathbf{P}_{\mathbf{A}^{\mathcal{S}}(\mathbf{x}^{\mathcal{S}})}$ and $\mathbf{P}_{\mathbf{A}^{\mathcal{T}}(\mathbf{x}^{\mathcal{T}})}$ with respect to $\theta_{\mathbf{A}^{\mathcal{S}}}$ and $\theta_{\mathbf{A}^{\mathcal{T}}}$. Then transferabilities of source samples are ranked according to the learned distance and a subset of $\widehat{n_{\mathcal{S}}}$ most transferable source instance are chosen to be used by the DANN component to learn domain-invariant features for target classification.

4.2.1 Approximate Wasserstein Distance

Following the approximation from 2.3.2, we define a data selection head that learn a function $h_{wass}^{\mathcal{ST}} : \mathbb{R}^d \to \mathbb{R}$ that estimate the Wasserstein-1 distance between source and target marginal representation distributions, which is formulated as:

$$d_{\mathcal{W}}\left(\mathbf{P}_{\mathbf{A}^{\mathcal{S}}(\mathbf{x}^{\mathcal{S}})}, \mathbf{P}_{\mathbf{A}^{\mathcal{T}}(\mathbf{x}^{\mathcal{T}})}\right) = \sup_{\|h_{wass}^{\mathcal{ST}}\| \leq 1} \mathbf{E}_{\mathbf{P}_{\mathbf{A}^{\mathcal{S}}(\mathbf{x}^{\mathcal{S}})}}\left[h_{wass}^{\mathcal{ST}}(\mathbf{x}^{\mathcal{S}})\right] - \mathbf{E}_{\mathbf{P}_{\mathbf{A}^{\mathcal{T}}(\mathbf{x}^{\mathcal{T}})}}\left[h_{wass}^{\mathcal{ST}}(\mathbf{x}^{\mathcal{T}})\right] \quad (4.7)$$

using the empirical Wasserstein-1 distance by maximizing the following loss with respect to θ_{wass}:

$$\mathcal{L}_{wass}^{\mathcal{ST}} = \frac{1}{n^s} \sum_{\mathbf{x} \sim \mathcal{S}} h_{wass}^{\mathcal{ST}}\left(\mathbf{A}^{\mathcal{S}}(\mathbf{x})\right) - \frac{1}{n^t} \sum_{\mathbf{x} \sim \mathcal{T}} h_{wass}^{\mathcal{ST}}\left(\mathbf{A}^{\mathcal{T}}(\mathbf{x})\right) \quad (4.8)$$

where n^s and n^t are number of source and target samples in a minibatch. For the above approximation to work, we need to enforce the Lipschitz constraint which will force the hypothesis class of $h_{wass}^{\mathcal{ST}}$ to be 1-Lipschitz. There are several ways to do this. First in Wasserstein-GAN ([ACB17]), it was suggested to use weight clipping on $h_{wass}^{\mathcal{ST}}$ after each update step. However, it was pointed out that this procedure may cause capacity underused and gradient vanishing or exploding problems. Therefore in the following work of [Gul+17], a more suitable approach was proposed for the task, in which a near-one constraint is put on the gradient norm of $h_{wass}^{\mathcal{ST}}$ with respect to its inputs. The motivation is that a differentiable function is 1-Lipschtiz if and only if it has gradients with norm at most 1 everywhere. Applying to our model, a gradient penalty $\mathcal{L}_{grad}^{\mathcal{ST}}$ is added to the loss $\mathcal{L}_{wass}^{\mathcal{ST}}$, resulting in the overall estimation problem for the Wasserstein distance as :

$$\max_{\theta_{wass}} \mathcal{L}_{wass}^{\mathcal{ST}} + \lambda_{gr} \mathcal{L}_{grad}^{\mathcal{ST}} \quad (4.9)$$

$$with \quad \mathcal{L}_{grad}^{\mathcal{ST}}(A) = \left(\left\|\nabla_A h_{wass}^{\mathcal{ST}}(A)\right\|_2 - 1\right)^2, \quad (4.10)$$

where λ_{gr} is a hyper-parameter which is responsible balancing the original estimation loss $\mathcal{L}_{wass}^{\mathcal{ST}}$ and the gradient penalty $\mathcal{L}_{grad}^{\mathcal{ST}}$.

4.2.2 Data Selection based on Wasserstein Distance

From previous analyses, it is clear that the effect of negative transfer can be serious in case of highly dissimilar domains. As our effective number of parameters used for adaptation is significantly fewer than that of the original BERT model, the data used for aligning distribution should also be denser - lower in quantity while higher

in quality. Therefore, we proposed to use a data selection mechanism based on the estimated Wasserstein distance. By minimizing the empirical distance using $\mathbf{A}^{\mathcal{S}}$ and $\mathbf{A}^{\mathcal{T}}$:

$$\min_{\theta_{\mathbf{A}^{\mathcal{S}}}, \theta_{\mathbf{A}^{\mathcal{T}}}} \mathcal{L}_{wass}^{\mathcal{ST}}, \tag{4.11}$$

we find the optimal representations that achieve the shortest transport distance between source and target samples. From there we select a subset of $\widehat{n_{\mathcal{S}}}$ source samples with the lowest $h_{wass}^{\mathcal{ST}}(\mathbf{A}^{\mathcal{S}}(\mathbf{x}))$ scores, which corresponds to $\widehat{n_{\mathcal{S}}}$ closest distances to target domain. These source instances will be used by the joint adapter $\mathbf{A}^{\mathcal{J}}$ in conjunction with target unlabelled data to learn domain-invariant features through the DANN component $h_{dann}^{\mathcal{J}}$.

There are several benefits of using Wasserstein distance as the proxy for data selection mechanism. First, Wasserstein distance is an instance of IPM-based approach (2.3), in which discrepancy between domains is computed taking into account the geometry of the actual data distributions. Thus, it is intuitive to pick source samples that are geometrically closed to samples from target distribution. Furthermore, Wasserstein distance is much weaker than KL-divergence on which DANN component based to update $\mathbf{A}^{\mathcal{J}}$ (an algorithm that aligns domains using KL-divergence also reducing the Wasserstein distance be the corresponding distributions , [ACB17]). Therefore, the minimal Wasserstein distance with respect to $\mathbf{A}^{\mathcal{S}}$ and $\mathbf{A}^{\mathcal{T}}$ should be proportional to the Wasserstein distance computed from the DANN-learned $\mathbf{A}^{\mathcal{J}}$.

4.3 Learning Domain-Invariance Features With Domain Separation Architecture

This section deals with components that are responsible for updating our main representation $\mathbf{A}^{\mathcal{J}}$. The joint adapter's representation will be directly used for downstream task, thus play a central role in improving performance across domains. To learn an representation that is as general as possible while also maintaining discriminative property, we proposed to combined two mechanisms with complementary effects - a relaxed domain-adversarial component and a domain disentanglement component.

4.3.1 Domain-Adversarial Neural Network (DANN)

Follow other methods that made used of divergence-based bound, we apply domain-adversarial training procedure to the representation of $\mathbf{A}^{\mathcal{J}}$. However, we propose multiple refinements to the original DANN approach in 2.3.1 to mitigate its flaws and learn better domain-invariant features as follow:

- **Dimension Reduction** It is known that discriminative features computed by high-level layers usually lie on low dimensional manifolds. As a result, naively applying DANN for BERT's representations, which require high dimension to capture contexts, can lead to gradient vanishing problem. We leverage the adapter's architecture to address this issue. Instead of the full layer's dimension output, we align domains based on the down-sampled version of the representations, computed by a_i^{down}, which only have dimension $c \ll d$.

- **Asymmetric Relaxation** [Wu+19] has shown that minimization the objectives of DANN can break down under two domains with shifting label distributions. Following their suggestion, we implement a simple relaxation on our domain classification objective:

$$\mathcal{L}^{\mathcal{J}}_{dann} = -\frac{1}{N}\sum_{i=1}^{N}\left[d_i\, log\left(\frac{h^{\mathcal{J}}_{dann}(\mathbf{A}^{\mathcal{J}}(\mathbf{x_i}))}{1+\beta}\right) + (1-d_i) log\left(1-\frac{h^{\mathcal{J}}_{dann}(\mathbf{A}^{\mathcal{J}}(\mathbf{x_i}))}{1+\beta}\right)\right] \text{ , } with\ \beta \geq 0 \tag{4.12}$$

  where d_i is domain label of samples x_i, $N = n^{\mathcal{S}} + n^{\mathcal{T}}$ is minibatch size, and β is a hyper-parameter controlling the strength of the relaxation. An simplified explanation for this modification is: consider a model that manage to learns to both classify source samples and align domains' marginal distributions perfectly. If there is a discrepancy between the label distributions of source and target domains (also refer to as target shift in 2.1), then it should be impossible for the model to classify target samples without errors because the output of model's classifier would be proportional to source domain's label distribution. This explanation also leads to an simple fix for a plausible optimal domain alignment, in which instead of trying to match source and target marginals completely, we only bound the maximal difference of the two distributions based on β:

$$\sup_{x\in\mathbf{X}} \frac{\mathbf{P}_{\mathcal{S}}(x)}{\mathbf{P}_{\mathcal{T}}(x)} \leq 1+\beta\,, \tag{4.13}$$

  If $\beta = 0$, then the relaxed version becomes the original formulation. [Wu+19] proved that using this method enable learning of domain-invariant features whether there is a label shift between domains or not.

- **Layer-wise Alignment** To enhance the alignment capability of our model, we apply domain-adversarial training on down-sampled versions of representations from every BERT's layers. In particular, we incorporate the previous relaxation to make up for the fact there is also a representation distribution shift across layers. Suppose same as final layers, every layers' representations are also used to predict labels, then each will encodes a different label distributions when projected into label space. These distributions will have varying information regarding true labels distributions, with lower layers ones, which representations encode quite broad knowledge, being more random while the highers, which representations are more task-specific, are closer to the true distributions, effectively reducing the possible amount of label shift between the two domains. Thus, it is intuitive to gradually reduce the strength of the relaxation going up in layers. We adopt the following annealing strategy:

$$\mathcal{L}^{\mathcal{J}}_{dann} = \sum_{i=0}^{11} \mathcal{L}^{\mathcal{J}}_{dann}\left(\mathbf{A}^{\mathcal{J}}_i, \beta_i, h^{\mathcal{J}}_i\right) \text{ , } with\ \beta_i = 2^{3-i}, \tag{4.14}$$

  where each term on the right-hand side of 4.14 is a different relaxed-DANN where $h^{\mathcal{J}}_i$ is a domain classifier taking layer i's representation $\mathbf{A}^{\mathcal{J}}_i(x)$ as input and using hyper-parameter β_i.

4.3.2 Domain Disentanglement based on Self-Supervised Task

To leverage the inductive bias of the three adapters architecture, we want the joint adapter $\mathbf{A}^{\mathcal{J}}$ to be a shared representation space containing common information between domains and no domain-specific information, while the domain-specific adapters $\mathbf{A}^{\mathcal{S}}$ and $\mathbf{A}^{\mathcal{T}}$ should only accommodate distinct knowledge that belong exclusively to their corresponding domain. Following the work of [LQH17] and [Bou+16], we impose an orthogonal constraint using the following similarity loss function:

$$\begin{aligned} \mathcal{L}_{sim}^{\mathcal{STJ}} &= \mathcal{L}_{sim}^{\mathcal{STJ}}\left(\mathbf{A}^{\mathcal{J}}, \mathbf{A}^{\mathcal{S}}\right) + \mathcal{L}_{sim}^{\mathcal{STJ}}\left(\mathbf{A}^{\mathcal{J}}, \mathbf{A}^{\mathcal{T}}\right) \\ \textit{where } \mathcal{L}_{sim}^{\mathcal{STJ}}\left(\mathbf{A}^{\mathcal{J}}, \mathbf{A}^{\mathcal{S}}\right) &= \|\mathbf{A}^{\mathcal{J}\top}\mathbf{A}^{\mathcal{S}}\|_F^2\,,\ \mathcal{L}_{sim}^{\mathcal{STJ}}\left(\mathbf{A}^{\mathcal{J}}, \mathbf{A}^{\mathcal{T}}\right) = \|\mathbf{A}^{\mathcal{J}\top}\mathbf{A}^{\mathcal{T}}\|_F^2 \end{aligned} \tag{4.15}$$

Minimizing $\mathcal{L}_{sim}^{\mathcal{STJ}}$ will force $\mathbf{A}^{\mathcal{J}}$ to be a complementary subspace of $\mathbf{A}^{\mathcal{S}}$ and $\mathbf{A}^{\mathcal{T}}$, resulting in independency. However, while $\mathbf{A}^{\mathcal{J}}$ is trained to be discriminative for the main task, $\mathbf{A}^{\mathcal{S}}$ and $\mathbf{A}^{\mathcal{T}}$ are basically free representations that are not constraint to any specific task. Thus, they can be pushed into trivial solutions where the network learns to map each representation into the same orthogonal space with $\mathbf{A}^{\mathcal{J}}$, not having any expressive capability of their corresponding domains. To address above problem, we incorporate a self-supervised component, using the popular Masked Language Model as our unsupervised task. The token predictor $h_{mlm}^{\mathcal{STJ}} : \mathbb{R}^d \leftarrow \mathbb{R}^V$, where V is the predefined vocabulary, is shared between source and target domains as we want the combined representation space is the same across domains:

$$\mathcal{L}_{mlm}^{\mathcal{STJ}} = \mathcal{L}_{mlm}^{\mathcal{STJ}}(\mathbf{A}^{\mathcal{JS}}) + \mathcal{L}_{mlm}^{\mathcal{STJ}}(\mathbf{A}^{\mathcal{JT}}) \tag{4.16}$$

$$\textit{where } \mathbf{A}^{\mathcal{JS}} = h_{mlm}^{\mathcal{STJ}}\left(\mathbf{A}^{\mathcal{J}}\left(\mathbf{x}^{\mathcal{S}}\right) + \mathbf{A}^{\mathcal{S}}\left(\mathbf{x}^{\mathcal{S}}\right)\right) \tag{4.17}$$

$$\textit{and } \mathbf{A}^{\mathcal{JT}} = h_{mlm}^{\mathcal{STJ}}\left(\mathbf{A}^{\mathcal{J}}\left(\mathbf{x}^{\mathcal{T}}\right) + \mathbf{A}^{\mathcal{T}}\left(\mathbf{x}^{\mathcal{T}}\right)\right) \tag{4.18}$$

Combining 4.15 and 4.16, the aim the Domain Separation component is to learn three representations $\mathbf{A}^{\mathcal{S}}$, $\mathbf{A}^{\mathcal{T}}$, $\mathbf{A}^{\mathcal{J}}$ such that the two pairs $(\mathbf{A}^{\mathcal{S}}, \mathbf{A}^{\mathcal{J}})$ and $(\mathbf{A}^{\mathcal{T}}, \mathbf{A}^{\mathcal{J}})$ are partition of source and target domains on contextual representation space learned from predicting masked tokens in a given context, through the minimization of the following objective:

$$\begin{aligned} \min_{\theta_{\mathbf{A}^{\mathcal{S}}}, \theta_{\mathbf{A}^{\mathcal{T}}}, \theta_{\mathbf{A}^{\mathcal{J}}}, \theta_{mlm}} \mathcal{L}_{sim}^{\mathcal{STJ}} + \mathcal{L}_{mlm}^{\mathcal{STJ}} = \\ \min_{\theta_{\mathbf{A}^{\mathcal{J}}}, \theta_{mlm}} \Bigg[\min_{\theta_{\mathbf{A}^{\mathcal{S}}}} \left(\mathcal{L}_{sim}^{\mathcal{STJ}}\left(\mathbf{A}^{\mathcal{J}}, \mathbf{A}^{\mathcal{S}}\right) + \mathcal{L}_{mlm}^{\mathcal{STJ}}\left(\mathbf{A}^{\mathcal{J}}, \mathbf{A}^{\mathcal{S}}\right) \right) \\ + \min_{\theta_{\mathbf{A}^{\mathcal{T}}}} \left(\mathcal{L}_{sim}^{\mathcal{STJ}}\left(\mathbf{A}^{\mathcal{J}}, \mathbf{A}^{\mathcal{T}}\right) + \mathcal{L}_{mlm}^{\mathcal{STJ}}\left(\mathbf{A}^{\mathcal{J}}, \mathbf{A}^{\mathcal{T}}\right) \right) \Bigg] \end{aligned} \tag{4.19}$$

The benefits of adding the MLM component is two-fold. On one hand, it serves as a constraint to learn an informative representations for domain-specific adapters. On the other hand, it also help conditioning joint adapter $\mathbf{A}^{\mathcal{J}}$ on unsupervised knowledge of unlabelled target data, which can have positive impact on the target domain's downstream task (similar to TAPT pretraining procedure from 3.2.1).

4.4 Learning Algorithms

Final Loss Taking it all together, our final training objective is given as:

$$\mathcal{L}_{total} = \mathcal{L}^{\mathcal{J}}_{task} + \lambda_d \mathcal{L}^{\mathcal{J}}_{dann} + \lambda_w \mathcal{L}^{\mathcal{ST}}_{wass} + \lambda_s \mathcal{L}^{\mathcal{ST}\mathcal{J}}_{sim} + \lambda_m \mathcal{L}^{\mathcal{ST}\mathcal{J}}_{mlm} \tag{4.20}$$

where $\lambda_d, \lambda_w, \lambda_s, \lambda_m$ are hyper-parameters which help to balance the importance of the corresponding loss with the main downstream loss. Of the five terms on the right-hand side of 4.20, $\mathcal{L}^{\mathcal{J}}_{dann}$ and $\mathcal{L}^{\mathcal{ST}}_{wass}$ are based on domain-divergence heads that requires a maximization step.

Alternating Minimization While the main task term and domain separation terms can be minimized following the standard gradient descent procedure, the other two terms require the maximization of the added heads in concurrent with the minimization of the adapters, resulting in a min-max optimization problem. Previous works that made used of DANN usually applied gradient reversal layer (GRL) to train the feature extractors, which only scaled the gradients flowing through by while leaving all other computations unchanged. We find this approach to be unstable, and following suggestions from [Goo+14] and [Shu+18], we design an alternating minimization algorithm that is compatible with our learning algorithm whilst also stabilizing the domain-adversarial training. The overall procedure is shown in shown in Algorithm 1 in which the min-max optimization problem is formulated as a two-stage process. In the first stage (Domain Step), all parameters are fixed except of the domain-adversarial heads. This step corresponds to approximation the divergences between domains' representations through maximizing the distance losses, which is equivalent to minimizing them using reverse domain labels (considering source samples come from target domain and and the other way round). After repeating this step k times (k is a hyper-parameter that controls the trade-off between computation and accuracy of the divergence estimations), a subset of source minibatch can be selected based on the approximated Wasserstein distance, which will be used to update joint adapter in next step. The following stage (Adapter Step) updates the rest of the previously fixed parameters using the standard gradient descent algorithm.

Inference At test time, a new sample $\mathbf{x}_{test}$ will go through the trained joint adapter $\mathbf{A}^{\mathcal{J}}$ to produce domain-invariant representation $\mathbf{A}^{\mathcal{J}}(\mathbf{x_{test}})$, which is then used by prediction head $h^{\mathcal{J}}_{task}$ to produce the corresponding event label.

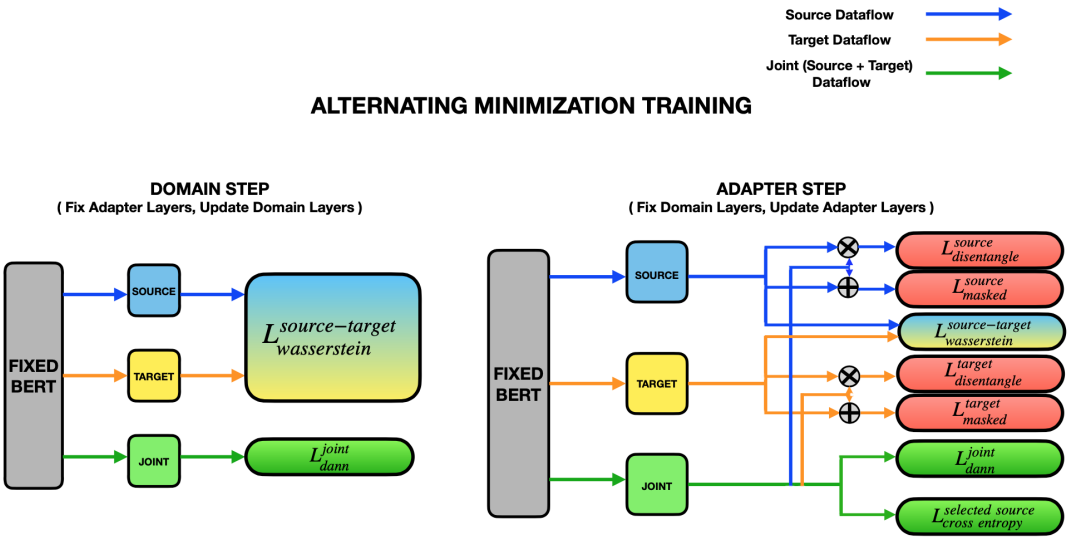


FIGURE 4.2: Alternating Minimization Training. In the left figure, Domain Step is repeated k times to obtain a good estimate for the current divergence between representation distributions of source and target domains. The right figure show the Adapter Step where the previously computed distances are used to update parameters of adapters and other downstream tasks.

Algorithm 1: Adapter-based finetuning for AHDS

Input: Source data $\mathcal{D}^{\mathcal{S}}_{\mathbf{XY}}$; target data $\mathcal{D}^{\mathcal{T}}_{\mathbf{X}}$; fixed BERT parameters θ_B ; minibatch size $N = n^{\mathcal{S}} + n^{\mathcal{T}}$ of $n^{\mathcal{S}}$ labelled source samples and $n^{\mathcal{T}}$ unlabelled target samples ; learning rate η ; number of domain step per adapter step k ; number of source instances selected $\widehat{n^{\mathcal{S}}}$

Output: Trained parameters $\theta_{\mathbf{A}^{\mathcal{S}}}, \theta_{\mathbf{A}^{\mathcal{T}}}, \theta_{\mathbf{A}^{\mathcal{J}}}, \theta_{task}, \theta_{dann}, \theta_{wass}, \theta_{mlm}$

Initialization: **identity_init**$(\theta_{\mathbf{A}^{\mathcal{S}}}, \theta_{\mathbf{A}^{\mathcal{T}}}, \theta_{\mathbf{A}^{\mathcal{J}}})$;
random_init$(\theta_{task}, \theta_{dann}, \theta_{wass}, \theta_{mlm})$

for *epoch* $= 1 \cdots N_{ep}$ **do**
  for *minibatch* $= 1 \cdots N_{mb}$ **do**
    Sample minibatch: $mb^{\mathcal{S}} = \left\{x_i^s, y_i^s\right\}_{i=1}^{n^{\mathcal{S}}}$ and $mb^{\mathcal{T}} = \left\{x_i^t\right\}_{i=1}^{n^{\mathcal{T}}}$ from $\mathcal{D}^{\mathcal{S}}_{\mathbf{XY}}$ and $\mathcal{D}^{\mathcal{T}}_{\mathbf{X}}$

    Domain Steps: update domain-divergence heads' parameters θ_{wass} and θ_{dann}, fix all other parameters
    for $t = 1 \cdots k$ **do**
      $g_w \leftarrow \nabla_{\theta_{wass}} \mathcal{L}^{\mathcal{ST}}_{wass}\left(mb^{\mathcal{T}}, mb^{\mathcal{S}}\right)$ ▷ `reverse domain labels`
      $g_d \leftarrow \nabla_{\theta_{dann}} \mathcal{L}^{\mathcal{J}}_{dann}\left(mb^{\mathcal{T}}, mb^{\mathcal{S}}\right)$ ▷ `reverse domain labels`
      $\theta_{wass} \leftarrow \theta_{wass} - \eta g_w$
      $\theta_{dann} \leftarrow \theta_{dann} - \eta g_d$
    end

    Data selection: Select subset of source minibatch with lowest distance to target minibatch:
    $\widehat{mb^{\mathcal{S}}} = argmin_{\widehat{mb^{\mathcal{S}}} \subset mb^{\mathcal{S}}} \mathcal{L}^{\mathcal{ST}}_{wass}\left(mb^{\mathcal{S}}, mb^{\mathcal{T}}\right).$

    Adapter Step: Fix parameters of domain-divergence heads, update adapters and other heads' parameters $\theta_{\mathbf{A}^{\mathcal{S}}}, \theta_{\mathbf{A}^{\mathcal{T}}}, \theta_{Aj}, \theta_{mlm}, \theta_{task}$.
    DSN head's gradients: using full source and target minibatchs
      $g_m \leftarrow \nabla_{\theta_{mlm}} \lambda_m \mathcal{L}^{\mathcal{ST}\mathcal{J}}_{mlm}\left(mb^{\mathcal{S}}, mb^{\mathcal{T}}\right)$
    Source classification head's gradients: using selected source minibatchs
      $g_t \leftarrow \nabla_{\theta_{task}} \mathcal{L}^{\mathcal{J}}_{task}(\widehat{mb^{\mathcal{S}}})$
    Adapters' gradient: using full source and target minibatchs
      $g_{as} \leftarrow \nabla_{\theta_{\mathbf{A}^{\mathcal{S}}}} \lambda_w \mathcal{L}^{\mathcal{ST}}_{wass} + \lambda_s \mathcal{L}^{\mathcal{ST}\mathcal{J}}_{sim} + \lambda_m \mathcal{L}^{\mathcal{ST}\mathcal{J}}_{mlm}$
      $g_{at} \leftarrow \nabla_{\theta_{\mathbf{A}^{\mathcal{T}}}} \lambda_w \mathcal{L}^{\mathcal{ST}}_{wass} + \lambda_s \mathcal{L}^{\mathcal{ST}\mathcal{J}}_{sim} + \lambda_m \mathcal{L}^{\mathcal{ST}\mathcal{J}}_{mlm}$
      $g_{aj} \leftarrow \nabla_{\theta_{\mathbf{A}^{\mathcal{J}}}} \mathcal{L}^{\mathcal{J}}_{task} + \lambda_d \mathcal{L}^{\mathcal{J}}_{dann} + \lambda_s \mathcal{L}^{\mathcal{ST}\mathcal{J}}_{sim} + \lambda_m \mathcal{L}^{\mathcal{ST}\mathcal{J}}_{mlm}$
    Parameters' updates
      $\theta_{\mathbf{A}^{\mathcal{S}}} \leftarrow \theta_{\mathbf{A}^{\mathcal{S}}} - \eta g_{as}$
      $\theta_{\mathbf{A}^{\mathcal{T}}} \leftarrow \theta_{\mathbf{A}^{\mathcal{T}}} - \eta g_{at}$
      $\theta_{\mathbf{A}^{\mathcal{J}}} \leftarrow \theta_{\mathbf{A}^{\mathcal{J}}} - \eta g_{aj}$
      $\theta_{mlm} \leftarrow \theta_{mlm} - \eta g_m$
      $\theta_{task} \leftarrow \theta_{task} - \eta g_t$
  end
end

Chapter 5

Experimental Studies

5.1 Tasks and Datasets

5.1.1 Event Identification

Timebank Dataset ([Pus+03]) Timebank is a fine-grained temporally annotated corpus of events and their positions and ordering in time. The text of the dataset were chosen from a wide range of sources from the news media domain: 12% are part of Document Understanding Conference corpus covering areas like biography, single and multiple events, 33% comes from transcribed broadcast news (ABC, CNN, PRI, VOA) and newswire (AP, NYT), and the rest are Wall Street Journal newswire texts.

Litbank Dataset ([SPB19]) Litbank is a recently introduced corpus of literary events that have taken place within the imaginary space of a novel. In contrast to previous works which mostly focus on the domain of contemporary news, events in literature are in a completely different spectrum of syntactic and semantic, from the board array of mental states to the strong emphasis on figurative language. The dataset gathers text from approximately the first 2000 words of 100 literary works from the Project Gutenberg corpus, with the goal of ensuring a rich variation of novelistic discourse so that samples are not only narratively and stylistically complex but also declarative and plot-driven.

Event Identification Task Given an event mention (usually a sentence or a sequence of words) and a position, the task is formulated as a binary classification problem in which the goal is to predict if the word at the position in the context of the mention is an event (positive) or not (negative).

5.1.2 Event Detection

Automatic Content Extraction 2005 (ACE-2005) dataset([WC05]) ACE-2005 is a densely annotated corpus collected from 6 different domains:

- Newswire (nw): 20% of the corpus; collected from AFP (Agence France Presse), APW (Associated Press), NYT (NewYork Times) and XIN (Xinhua News Agency).
- Broadcast news (bn): 20% of the corpus; collected from CNN (Cable News Network)and CNNHL (CNN Headline News).
- Broadcast conversation (bc): 15% of the corpus; collected from CNN-CF (CNN CrossFire), CNN-IP (CNN Inside Politics) and CNN-LE (CNN Late Edition).

- Weblog (wl): 15% of the corpus; collected from various internet weblogs and bulletin boards.
- Usenet Newsgroups (un): 15% of the corpus; collected from
- Conversational Telephone Speech (cts): 15% of the corpus; collected from EARS (Fisher 2004 Telephone Speech Collection Supplement).

Events of ACE-2005 dataset are fine-grainedly labeled into 33 types, each consists of 8 different sub-types. They comes from a targeted data selection procedure which algorithmically picks documents from a large pool to maximize the overall count of each type/sub-type.

Event Detection Task Given an event mention (usually a sentence or a sequence of words) and a position, the task is formulated as a binary classification problem in which the goal is to predict if the word at the position in the context of the mention belongs to one of the predefined 33 event labels (positive) or not (negative).

5.2 Baselines and Implementation Details

Baselines To demonstrate the effectiveness of our proposed solution with several other baselines. In particular, for the task of event identification, we compare our model AHDS with the performance of four models implemented in [NR20] , three of which are $LSTM$, $BiLSTM$, $BERT$ that did not use any alignment mechanism to address the unsupervised domain adaptation setting, while the last model $BERT + DANN$ follow a naive domain-adversarial training procedure based on a pretrained BERT-base model. Regarding the event detection task, our baselines include:

- **Previous Works**: We use the reported results of previous systems $MaxEnt$, $Joint + Local$, $B - RNN$, $NC - CNN$, $SELF$ in the original works ([LJH13], [NG15], [NCG16], [NG16], [Hon+18]). These models represent approaches that do not make use of contextual embeddings.
- **BERT**: We finetune a pretrained BERT-base model on only in-domain dataset
- **BERT+DANN**: We implement a naive domain-adversarial training procedure and re-train a pretrained BERT-base model using data from both domains.
- **BERT+Adapter**: To demonstrate the ability of adapter approach to retain original performance, we adopt AbF on a pretrained BERT-base model and finetune only the added adapter modules on source dataset.

Experiment Settings To formulate the unsupervised domain adaptation setting from the origin dataset of each task, we split the test corpus into two parts at the ratio of 1 to 4, a training target domain dataset which models have access to without labels when learning, and a test target domain dataset that models are evaluated upon. In particular:

- Event Identification: Transfer experiments are performed in two ways: Litbank-to-Timebank, and the reserve direction, Timebank-to-Litbank. The data statistics of each setting are presented in tables 5.1a and 5.1b. We can see that each direction represents a different adaptation situation between high and low event density.

Statistics	In-Domain (Train-Litbank)	Out-of-Domain (Train-Timebank)	Out-of-Domain (Test-Timebank)
#Docs	100	37	146
#Tokens	210532	20047	77216
#Events	7849	3268	12931
Event Density	3.73%	16.30%	16.74%

(A) Timebank to Litbank

Statistics	In-Domain (Train-Timebank)	Out-of-Domain (Train-Litbank)	Out-of-Domain (Test-Litbank)
#Docs	183	20	80
#Tokens	97263	41962	168570
#Events	16199	1778	6071
Event Density	16.65%	4.23%	3.60%

(B) Litbank to Timebank

Statistics	bn + nw	bc		cts		wl		un	
	Train	Train	Test	Train	Test	Train	Test	Train	Test
#Docs	264	12	48	8	31	24	95	10	39
#Samples	38644	3130	64590	2885	10972	3424	12767	3214	11678
#Events	2045	144	665	82	327	78	386	101	516
Chi-Square	0.00	75.01	204.47	90.37	339.23	140.07	346.36	452.53	372.73

(C) ACE-05

TABLE 5.1: Experiment Settings. Figure (a) is Timebank to Litbank adaptation task. Figure (b) is Litbank to Timebank adaptation task. Figure (c) is ACE-05 adaptation task.

- Event Detection: To create a sizeable source domain dataset for training, we gather data from two closely related domains, bn and nw. Then each of the other domains is considered the target domain of a single adaptation setting, where 20% of its documents are used as unlabelled target data and the other 80% are considered as test dataset. The overall statistics of every settings are listed in table 5.1c. From the Chi-squared statistic computed between source and target domains' label categorical distributions, given that the critical value is 45.00, it is apparent that there are significant shifts among domains with which models need to deal for a good target performance.

5.3 Main Experimental Results

5.3.1 Event Identification

The results of our event identification experiments are presented in tables 5.3.1 and 5.3.1. From Table 5.3.1 (transfer from Litbank to Timebank), our proposed model AHDS improve out-of-domain performance for all models, outperforming naive adoption of DANN on BERT by more than 10 points in F1. At Following a similar trends, Table 5.3.1 (transfer from Timebank to Litbank) shows AHDS reaching an F1 score of 61.1, a 11.5 points gap considering the best baseline performance. We also note that high precision is observed from models transferred from Litbank to Timebank, while the other direction counterparts have high recall. An explanation for this phenomenon is that models are tend to be conservative (high precision) when transferring from lower event density corpus Litbank, and are less so (high recall) if source domain Timebank has higher event density. Therefore, a desired property of a good model is be able to balance between the two statistics, which our model is able to achieve and is thus more general across domains in comparison to the baselines.

Event Detection

Table 5.3.1 showcases he results of our event detection experiments. The main conclusions from the table include:

Models	In-Domain(Litbank)			Timebank		
	P	Q	F1	P	R	F1
LSTM	70.7	78.4	74.4	23.5	75.2	35.8
BiLSTM	75.4	76.3	75.9	27.6	68.8	39.4
BERT	79.6	84.3	81.9	28.1	84.8	42.2
BERT+DANN	79.8	85.6	82.6	30.3	80.8	44.1
AHDS (Our model)	90.9	88.4	**89.6**	40.0	81.3	**53.6**

TABLE 5.2: Model performance on unsupervised domain transfer for event identification experiments from Timebank to Litbank

Models	In-Domain(Litbank)			Timebank		
	P	Q	F1	P	R	F1
LSTM	61.9	61.5	61.7	86.1	17.1	28.5
BiLSTM	64.5	61.7	63.1	91.8	14.4	24.9
BERT	73.5	72.7	73.1	88.1	28.2	42.7
BERT+DANN	71.9	71.3	71.6	85.0	35.0	49.6
AHDS (Our model)	77.7	75,65	**76.7**	83.2	48.5	**61.1**

TABLE 5.3: Model performance on unsupervised domain transfer for event identification experiments from Litbank to Timebank

Models	In-Domain(bn+nw)			bc			cts			wl			un		
	P	Q	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
MaxEnt	74.6	59.4	66.0	70.1	54.5	61.3	66.4	49.9	56.9	59.4	34.9	43.9	—	—	—
Joint+Local	73.5	62.7	67.7	70.3	57.2	63.1	64.9	50.8	57.0	59.5	38.4	46.7	—	—	—
B-RNN	71.4	63.5	67.1	70.7	62.1	66.1	70.0	54.5	61.0	52.7	38.3	44.2	66.2	46.0	54.1
NC-CNN	74.9	66.5	70.4	73.6	64.7	68.8	71.7	57.3	63.6	57.8	40.3	47.4	71.7	49.0	58.1
SELF	73.8	65.7	69.5	70.0	67.2	68.9	68.3	60.2	63.3	58.0	44.0	50.0	—	—	—
BERT	77.5	77.5	77.5	75.2	71.8	73.5	75.1	69.9	72.4	70.2	57.9	60.6	68.9	67.5	68.2
BERT+DANN	77.4	75.8	76.6	72.8	69.4	70.9	73.4	39.9	51.2	69.2	50.5	58.4	68.8	59.2	63.6
BERT+Adapter	76.8	76.2	76.7	78.5	72.9	75.6	77.3	69.5	73.2	64.3	56.9	60.3	72.4	69.0	70.6
AHDS (Our model)	79.7	75.7	**77.7**	78.5	75.6	**76.9**	78.4	73.2	**75.6**	66.2	60.3	**63.1**	73.5	71.3	**72.3**

TABLE 5.4: Model performance on unsupervised domain adaption for event detection experiments among domains of ACE-05

- The baseline systems $MaxEnt$, $Joint + Local$, $B - RNN$, $NC - CNN$, $SELF$ achieve high performance on the source domain, but degrade dramatically on all target domains due to the dataset shift.
- The BERT baselines manage to surpass all previous non-contextual baselines, without using any mechanism to address the discrepancy among domains. This is due to the generalization potential of large unsupervised pretrained language model. However, naively adopting DANN into BERT has an adverse effects, notably reducing the performance of $BERT + DANN$ on all target domains. This outcome further emphasizes the need for a compatible implementation method for domain-adversarial training on BERT's representations.
- The results of $BERT + Adapter$ prove that adapter-based finetuning procedure is not only able to retain performance, but also prevent overfitting through capacity reduction, thus performing better than finetuned $BERT$ models in case where they follow too closely to source domain.
- Finally and most importantly, our proposed model AHDS consistently achieves the best adaptation performance across all the target domains. In settings where domains are closely related such as bc and cts, AHDS is more general and thus performs better on target domain. On the other hand, AHDS significantly outperforms baselines (3 to 5 points increase in F1 score) when transferring to target domains that are highly diverged from source domains (wl and un).

5.4 Ablation Studies

In this section, we provide thorough examination of our suggested framework. First, ablation studies on the indivivdual effect of each component is presents. We then evaluate our most important component (DANN) by comparing our proposed implementation with other possible settings.

5.4.1 Component-wise Analysis

To examine the effect of each of the proposed component with respect to all of the others, We perform an extensive ablation analysis for our AHDS model by measuring domain adaptation performance of each trained instance, with a single main

Models	In-Domain(bn+nw)			bc			cts			wl			un		
	P	Q	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
AHDS	79.7	75.7	**77.7**	78.5	75.6	**76.9**	78.4	73.2	**75.6**	66.2	60.3	**63.1**	73.5	71.3	**72.3**
AHDS-D	75.8	79.7	77.7	74.4	76.8	75.5	78.4	70.0	73.9	67.2	57.1	61.7	71.3	70.3	70.8
AHDS-W	79.2	76.4	77.7	77.8	73.5	75.6	80.8	70.1	75.1	70.2	53.7	60.9	74.0	66.7	70.1
AHDS-M	78.1	76.5	77.3	80.4	71.0	75.4	77.0	69.4	73.3	68.9	55.1	61.2	73.5	68.6	71.0
AHDS-S	77.5	77.1	77.3	78.7	72.4	75.4	79.2	70.7	74.7	65.5	57.9	61.5	72.7	67.9	70.2
AHDS-last	76.2	78.5	77.3	79.6	72.6	75.9	79.7	69.4	74.2	66.4	56.6	61.0	73.0	69.7	71.3

TABLE 5.5: Ablation Study Results

component removed, on ACE-05. In table 5.5, each row corresponds to these following architectures:

- AHDS: Our full model, with 4 main domain adaptation components - adapters, data selection, layer-wise domain critic, domain separation (masked LM and similarity constraint).
- $AHDS - D$: Our full model with domain critic component removed. Classification uses representations of $\mathbf{A}^{\mathcal{J}}$, which are only constrained by DSN components, on a selected subset of source samples.
- $AHDS - W$: Our full model with data selection component removed. Classification uses representations of $\mathbf{A}^{\mathcal{J}}$ on all of source samples.
- $AHDS - M$: Our full model with masked LM component removed. Classification uses representations of $\mathbf{A}^{\mathcal{J}}$, which are not combined with $\mathbf{A}^{\mathcal{S}}$ and $\mathbf{A}^{\mathcal{T}}$ for the unsupervised task, on a selected subset of source samples.
- $AHDS - S$: Our full model with similarity constraint component removed. Classification uses representations of $\mathbf{A}^{\mathcal{J}}$, which are not disentangled from $\mathbf{A}^{\mathcal{S}}$ and $\mathbf{A}^{\mathcal{T}}$, on a selected subset of source samples.
- $AHDS - last$: Our full model, but the domain-adversarial training is only applied on the representations of the last layer.

The results from our ablation studies show that the incomplete models performed consistently worse compare to the full model. In particular, while the in-domain performances are retained across settings, different domains experience different change in performance depend on its relation with the source domain. In bc domain where the dataset shift is not significant, the figures for the partial models are more or less the same. However, as the full model inherits the positive impact of every components, it is more general and thus performs better in target domain. On the other hand, datasets drawn from the domains of wl and un, which are substantially diverged from the source domain. Thus, components that address the domains' dissimilarity play important roles in improve target performance, and models such as DANN-W and DANN-D have the lowest results. We also validate the benefit of our layer-wise alignment in DANN component compare to the naive approach where only the final representations is used for the domain-adversarial procedure. We provide more detail analyses on the impact of different layers' representations on the ability to extract domain-invariant features of the model in the next section.

5.4.2 DANN Analysis

The central component of our architecture is undoubtedly DANN as other components work together to provide a good condition for DANN to jointly extract domain-invariant features and classify events. Thus, finding the optimal way to implement DANN for BERT is also an important question that needs further investigation. This section aims to demonstrate the effect of our multi-layer implementation of DANN. In particular, we apply alignment to different portions of BERT's layers (each contains vary degree of context information, from syntactic to semantic). Specifically, we consider the following strategies for choosing which part of BERT layers to go through DANN :

- Lower Layers Adaptation (*Lower*): $\mathbf{A}^{\mathcal{J}}$'s outputs from layers 0 to 3 are aligned.
- Middle Layers Adaptation (*Middle*): $\mathbf{A}^{\mathcal{J}}$'s outputs from layers 4 to 7 are aligned.
- Higher Layers Adaptation (*High*): $\mathbf{A}^{\mathcal{J}}$'s outputs from layers 8 to 11 are aligned.
- Last Layers Adaptation (*Last*): $\mathbf{A}^{\mathcal{J}}$'s final layer's output is aligned.
- Full Layers Adaptation (*Full*): the suggested implementation - $\mathbf{A}^{\mathcal{J}}$'s outputs from all 12 layers are aligned.
- Full Layers Adaptation using up-sampled representation (Up): same as the suggested implementation except the representation with full dimension (768) is used instead of the down-sampled ones with lower dimension (96).
- Full Layers Adaptation without Relaxation ($Full - Rel$): same as the suggested implementation except no relaxation is applied on loss functions.

Table 5.4.2 showcases the results of our experiment on different domain-adversarial settings. Overall, we observe that performance degrades on all three partial adaptation setting Lower-Middle-Higher. However, the changes vary across domains in each situation, probably stemming from the fact that different layer's representations contain distinct type of contextual information - from general to specific, including symbolic, syntatic, and semantic knowledge, therefore alignment address different types of dataset shifts. On the other hand, taking only the last layer's representations as input for DANN component, as the naive domain-adversarial procedure, performs worse compare to all other multi-layer counterparts. Furthermore, using full dimension representation significantly reduces generalize capability to target domains of model. This result proves the benefit of bottleneck architectures, not only the alignment of down-sampled representations more effective, but the free parameters of up-sampled project layers also increase model's capacity for the main downstream task. Last but not least, the figure for settings where there is no relaxation applied also demonstrate the consistent usefulness of this modification on the overall model.

Models	bc			cts			wl			un		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Lower	77.5	73.9	75.6	74.1	74.7	74.4	67.5	56.7	61.6	73.0	69.4	71.0
Middle	78.1	73.4	75.7	78.5	70.4	74.2	66.4	56.7	61.2	71.6	70.8	71.2
Higher	79.2	73.3	76.1	77.5	71.2	74.2	67.4	55.5	60.9	72.9	68.1	70.4
Last	79.6	72.6	75.9	79.7	69.5	74.2	66.4	56.6	61.0	73.0	69.7	71.3
Up	77.6	73.2	75.3	74.5	71.7	73.0	66.6	53.1	59.1	69.0	67.7	68.4
Full	78.5	75.6	**76.9**	78.4	73.2	**75.6**	66.1	59.6	**62.6**	73.5	71.3	**72.3**
Full-Rel	77.8	75.1	76.5	76.1	74.2	75.2	67.2	58.0	62.3	72.2	70.6	71.4

TABLE 5.6: DANN-Layers Analysis Results

Chapter 6

Conclusion

6.1 Summary of Findings

We propose an effective architecture and algorithm for the task of text classification in unsupervised domain adaptation setting. Our model (AHDS) satisfies the key properties that we put forward in the beginning of this thesis as follow:

- Domain Adaptation: AHDS makes multiple attempts to align the marginal distributions between domains, first finds and remove from adaptation process source samples that are too distinct from target domain, then apply domain-adversarial training at multiple level of representations.
- Architecture Augmentation: AHDS uses pretrained BERT model as the base of AbF process. Therefore, our model is able to leverage the representation power of the large LM.
- Generalization Performance: AHDS is designed with the central goal of creating a joint representation that is able to generalize to both domains. Experimental results shows that our model manage to outperform both baseline and state-of-the-art approaches.
- Computational Efficiency: AHDS is built upon a fixed pretrained BERT using adapter modules which added in total less than 10% the parameters number of the original model. Thus, AHDS is both space and time efficient.
- Task-agnostic and Domain-agnostic: AHDS's training procedure is both domain-agnostic and task-agnostic. As a result, it is easy to adapt AHDS to many other learning settings.

6.2 Future Directions

Using target-specific representation: One of the limitations of DSN architecture is that target adapter's representations that contains beneficial information for model's target performance are unused. Thus, providing access to them for the task classifier could potentially lead to improvement. Furthermore, newer self-attention based architectures could be used in placed the premature-developed BERT can also boost both accuracy and computational efficiency of the overall framework. Last but not least, the MLM component is, though necessary, an inefficient unsupervised task and provides marginal improvement. Other self-supervised methods with better efficiency and capability have been proposed ([Cla+20]) and should be suitable replacements.

Regularization Another line of research ([Sai+17], [Shu+18], [Kum+18]) on domain adaptation focus on the regularization of model's architecture and representation to enhance model's generalization capability. These methods have complementary effect with distribution alignment approaches, thus can be used jointly to further improve model's performance.

Other Tasks and Domains: AHDS is a general framework that is task-agnostic and domain-agnostic. Thus, it is possible to leverage AHDS's strong distribution alignment capability on corpora from various domains, especially low-resource ones with very distinct marginal distributions such as medical, biology,... Furthermore, AHDS can also be extended to harder and more general NLP tasks such as sequence classification and sequence tagging.

More general adaptation setting: Our current experiments are conducted on the basic setting of UDA, in which there are one source domain and one target domain of same task. The nature of our algorithm is not only domain-agnostic and task-agnostic, but also efficient with respect to large LM. Thus, it is possible to adapt our current framework to more general setting of varying datasets, domains and tasks such as multi-source domain adaptation and domain generalization.

Bibliography

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan". In: *arXiv preprint arXiv:1701.07875* (2017).

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[Ben+07] Shai Ben-David et al. "Analysis of representations for domain adaptation". In: *Advances in neural information processing systems*. 2007, pp. 137–144.

[Ben+10] Shai Ben-David et al. "A theory of learning from different domains". In: *Machine learning* 79.1-2 (2010), pp. 151–175.

[Bou+16] Konstantinos Bousmalis et al. "Domain separation networks". In: *arXiv preprint arXiv:1608.06019* (2016).

[Cha+19] Sneha Chaudhari et al. "An attentive survey of attention models". In: *arXiv preprint arXiv:1904.02874* (2019).

[Cla+20] Kevin Clark et al. "Electra: Pre-training text encoders as discriminators rather than generators". In: *arXiv preprint arXiv:2003.10555* (2020).

[Dev+18] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[Gan+16] Yaroslav Ganin et al. "Domain-adversarial training of neural networks". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 2096–2030.

[Goo+14] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2014), pp. 139–144.

[Gul+17] Ishaan Gulrajani et al. "Improved training of wasserstein gans". In: *Advances in neural information processing systems*. 2017, pp. 5767–5777.

[Gur+20] Suchin Gururangan et al. "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks". In: *arXiv preprint arXiv:2004.10964* (2020).

[Hon+18] Yu Hong et al. "Self-regulation: Employing a generative adversarial network to improve event detection". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 515–526.

[Hou+19] Neil Houlsby et al. "Parameter-efficient transfer learning for NLP". In: *arXiv preprint arXiv:1902.00751* (2019).

[KF14] Meelis Kull and Peter A. Flach. "Patterns of dataset shift". In: 2014.

[Kum+18] Abhishek Kumar et al. "Co-regularized alignment for unsupervised domain adaptation". In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 9345–9356.

[LJH13] Qi Li, Heng Ji, and Liang Huang. "Joint event extraction via structured prediction with global features". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2013, pp. 73–82.

[LP20] Pirmin Lemberger and Ivan Panico. *A Primer on Domain Adaptation*. 2020. arXiv: 2001.09994 [cs.LG].

[LQH17] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. "Adversarial multi-task learning for text classification". In: *arXiv preprint arXiv:1704.05742* (2017).

[NCG16] Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. "Joint event extraction via recurrent neural networks". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016, pp. 300–309.

[NG15] Thien Huu Nguyen and Ralph Grishman. "Event detection and domain adaptation with convolutional neural networks". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. 2015, pp. 365–371.

[NG16] Thien Huu Nguyen and Ralph Grishman. "Modeling skip-grams for event detection with convolutional neural networks". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 886–891.

[NR20] Aakanksha Naik and Carolyn Rosé. "Towards Open Domain Event Trigger Identification using Adversarial Domain Adaptation". In: *arXiv preprint arXiv:2005.11355* (2020).

[Pfe+20] Jonas Pfeiffer et al. "AdapterFusion: Non-destructive task composition for transfer learning". In: *arXiv preprint arXiv:2005.00247* (2020).

[Pus+03] James Pustejovsky et al. "The timebank corpus". In: *Corpus linguistics*. Vol. 2003. Lancaster, UK. 2003, p. 40.

[Red+19] Ievgen Redko et al. *Advances in Domain Adaptation Theory*. Elsevier, Aug. 2019. URL: https://hal.archives-ouvertes.fr/hal-02286281.

[RKR21] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. "A primer in bertology: What we know about how bert works". In: *Transactions of the Association for Computational Linguistics* 8 (2021), pp. 842–866.

[Rud19] Sebastian Ruder. "Neural Transfer Learning for Natural Language Processing". In: (2019). URL: http://ruder.io/thesis/neural_transfer_learning_for_nlp.pdf.

[Sai+17] Kuniaki Saito et al. "Adversarial dropout regularization". In: *arXiv preprint arXiv:1711.01575* (2017).

[She+18] Jian Shen et al. "Wasserstein distance guided representation learning for domain adaptation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[Shu+18] Rui Shu et al. "A dirt-t approach to unsupervised domain adaptation". In: *arXiv preprint arXiv:1802.08735* (2018).

[SPB19] Matthew Sims, Jong Ho Park, and David Bamman. "Literary event detection". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 3623–3634.

[Vas+17] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[WA20] Dustin Wright and Isabelle Augenstein. "Transformer Based Multi-Source Domain Adaptation". In: *arXiv preprint arXiv:2009.07806* (2020).

[WC05] C. Walker and Linguistic Data Consortium. *ACE 2005 Multilingual Training Corpus*. LDC corpora. Linguistic Data Consortium, 2005. ISBN: 9781585633760. URL: https://books.google.com.vn/books?id=SbjjuQEACAAJ.

[Wu+19] Yifan Wu et al. "Domain adaptation with asymmetrically-relaxed distribution alignment". In: *arXiv preprint arXiv:1903.01689* (2019).

[Zha+20] Sicheng Zhao et al. *Multi-source Domain Adaptation in the Deep Learning Era: A Systematic Survey*. 2020. arXiv: 2002.12169 [cs.LG].

[Zol83] Vladimir Mikhailovich Zolotarev. "Probability metrics". In: *Teoriya Veroyatnostei i ee Primeneniya* 28.2 (1983), pp. 264–287.

[ZZY12] Chao Zhang, Lei Zhang, and Jieping Ye. "Generalization bounds for domain adaptation". In: *Advances in neural information processing systems* 25 (2012), pp. 3320–3328.