Domain adaptation approaches for classifying social media crisis data

by

Hongmin Li

B.A., Nanjing University of Posts and Telecommunications, China, 2007

M.A., Nanjing University of Posts and Telecommunications, China, 2010

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2021

Abstract

Social media platforms such as Twitter provide valuable information for aiding first response during emergency events. Machine learning could be used to build automatic tools for filtering and categorizing useful information from the flood of information posted by eyewitnesses during a disaster. However, supervised learning algorithms rely on labeled data, which is not readily available for an emerging target disaster. While labeled data might be available for a prior source disaster (or a set of prior source disasters), supervised classifiers learned only from the source disaster(s) may not perform well on the target disaster, as each event has unique characteristics (e.g., type, location, culture), and may cause different social media responses. Therefore, domain adaptation approaches, which address the above limitation by learning classifiers from unlabeled target data in addition to source labeled data, represent a promising direction for social media crisis data classification tasks.

This thesis focuses on disaster tweet classification tasks, including classification of tweets as relevant to a disaster or not relevant, and classification of tweets as informative to disaster response teams or not informative. In the single-source setting, we propose several domain adaptation approaches for such tasks. More precisely, we first propose approaches based on Expectation Maximization and Self-training, performed on top of supervised Naive Bayes classifiers to classify tweets in categories of interest. We also employ a feature adaptation method (called Correlation Alignment) and combine it with Self-training to train weighted Naive Bayes classifiers. Experimental results on the task of identifying tweets relevant to a disaster of interest show that the domain adaptation classifiers are better as compared to the supervised baselines learned only from labeled source data.

In addition to the single-source setting, we also consider a multi-source setting, where several source disasters are used to transfer knowledge to a target disaster. Under the multi-source domain adaptation setting, we evaluate how different representations based on pre-trained word embeddings and sentence encoding models perform when used with supervised classifiers. The word-embeddings are pre-trained on very large unlabeled corpora, and can thus capture semantic information (e.g., similar words are close in the embedding space). We use the pre-trained word embeddings and sentence encoding models to design simple but effective representation-based adaptation approaches for disaster tweet classification. We further apply the Self-training approach on top of these models, and obtain domain adaptation models that are shown experimentally to perform better than the supervised models on the task of identifying relevant versus irrelevant tweets. We also train crisis specific word embeddings with our own crisis tweet corpora. The resulting embeddings can be used for a variety crisis tweet classification tasks. Finally, we design domain adaptation models on top of state-of-the-art pre-trained language models (e.g., BERT) for social media crisis data classification, and show the effectiveness of such model for disaster tweet classification.

This thesis contributes to the crisis informatics research by introducing domain adaptation approaches for social media crisis data classification. The proposed approaches have the potential to be used in practice and help with the information overload problem that disaster response teams face currently.

Domain adaptation approaches for classifying social media crisis data

by

Hongmin Li

B.A., Nanjing University of Posts and Telecommunications, China, 2007

M.A., Nanjing University of Posts and Telecommunications, China, 2010

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2021

Approved by:

Major Professor
Doina Caragea

Copyright

Abstract

Social media platforms such as Twitter provide valuable information for aiding first response during emergency events. Machine learning could be used to build automatic tools for filtering and categorizing useful information from the flood of information posted by eyewitnesses during a disaster. However, supervised learning algorithms rely on labeled data, which is not readily available for an emerging target disaster. While labeled data might be available for a prior source disaster (or a set of prior source disasters), supervised classifiers learned only from the source disaster(s) may not perform well on the target disaster, as each event has unique characteristics (e.g., type, location, culture), and may cause different social media responses. Therefore, domain adaptation approaches, which address the above limitation by learning classifiers from unlabeled target data in addition to source labeled data, represent a promising direction for social media crisis data classification tasks.

This thesis focuses on disaster tweet classification tasks, including classification of tweets as relevant to a disaster or not relevant, and classification of tweets as informative to disaster response teams or not informative. In the single-source setting, we propose several domain adaptation approaches for such tasks. More precisely, we first propose approaches based on Expectation Maximization and Self-training, performed on top of supervised Naive Bayes classifiers to classify tweets in categories of interest. We also employ a feature adaptation method (called Correlation Alignment) and combine it with Self-training to train weighted Naive Bayes classifiers. Experimental results on the task of identifying tweets relevant to a disaster of interest show that the domain adaptation classifiers are better as compared to the supervised baselines learned only from labeled source data.

In addition to the single-source setting, we also consider a multi-source setting, where several source disasters are used to transfer knowledge to a target disaster. Under the multi-source domain adaptation setting, we evaluate how different representations based on pre-trained word embeddings and sentence encoding models perform when used with supervised classifiers. The word-embeddings are pre-trained on very large unlabeled corpora, and can thus capture semantic information (e.g., similar words are close in the embedding space). We use the pre-trained word embeddings and sentence encoding models to design simple but effective representation-based adaptation approaches for disaster tweet classification. We further apply the Self-training approach on top of these models, and obtain domain adaptation models that are shown experimentally to perform better than the supervised models on the task of identifying relevant versus irrelevant tweets. We also train crisis specific word embeddings with our own crisis tweet corpora. The resulting embeddings can be used for a variety crisis tweet classification tasks. Finally, we design domain adaptation models on top of state-of-the-art pre-trained language models (e.g., BERT) for social media crisis data classification, and show the effectiveness of such model for disaster tweet classification.

This thesis contributes to the crisis informatics research by introducing domain adaptation approaches for social media crisis data classification. The proposed approaches have the potential to be used in practice and help with the information overload problem that disaster response teams face currently.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my deep gratitude to Professor Doina Caragea, a wonderful research advisor and a good friend and mentor who is always encouraging whenever I am in self-doubt. None of the work in this thesis would be possible without her patient guidance, encouragement and inspiring advice. I would not have come this far in this journey without her support and help both in research work and personal life.

I would also like to thank Professor Cornelia Caragea for the initial collaboration that started this research. I am also grateful for my colleagues, Nic Herdon and Nicolais Guevara, for collaborations in the social media crisis data analysis project and other projects. I want to thank Professor Torben Amtoft, Professor Dan Andresen and Professor Caterina Scoglio for serving as my committees and providing constructive suggestions and feedback along the development of this thesis.

I would also like to thank Joelle Pitts from Hale Library for the research assistant opportunity in New Literacies Alliance project that provided financial support for my first year of PhD study. I want to thank our department head Professor Scott DeLoach, as well as other professors in the department, for trusting me to be their teaching assistant. I also want to thank many of the professors of Computer Science Department for their great teaching, and also to the Beocat staff in the department for providing help to run the experiments in this research.

I would like to thank all my friends at KSU for their generous help, especially Yan Kuang, Huang Zhu, Rui Zhuang, Yang Xue and Hong Liu. They have helped me go through many difficulties through my PhD journey and brought fun to my life.

I would also like to thank my friend Doug Stein for proof-reading part of the thesis.

Finally, I wish to thank my family for their support and encouragement throughout my study, especially my husband Zhi, who have enabled me to start this journey, supported me and encouraged me in his own way when I needed it.

Dedication

To my parents and my son Luke.

Notation

x	Scalar
$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
X	Random variable or random vector
Y	Random variable or random vector
P	Probability mass function
$P(X)$	Probability distribution over a discrete random variable or over a variable whose type has not been specified
p	Probability density function
$p(X)$	Probability distribution over a continuous variable
θ	Model parameters or probability parameters
$\mathbf{W}$	Weight matrix
$\mathcal{L}$	Loss function
$\mathcal{T}$	Task
$\mathcal{D}$	Domain

Chapter 1

Introduction

1.1 Motivation

Text classification is one of the fundamental tasks in machine learning (ML) and Natural Language Processing (NLP). It has many real-life applications, including spam email filtering, news topic categorization and sentiment analysis, as well as recent applications in social media crisis data analysis. Usually, text classification is treated as a supervised machine learning problem, where we provide labeled training examples (e.g., positive or negative texts in sentiment analysis) as input, and use the examples to learn a model that maps inputs to labels, and can be used to label future unseen data. In order to train a supervised machine learning model that generalizes well to future unseen test data, the model should be trained on a large training dataset, and the test data should have the same distribution as the training data. But these conditions do not always hold true in real world due to two main challenges. The first challenge comes from the fact that labeled data is too expensive and time-consuming to obtain. For example, annotating a large amount of data for an on-going event may be prohibitive because time is critical during emergencies. The second challenge comes from the fact that data distribution may shift over time. For example, in the context of disasters, things like damage or affected areas evolve quickly and result in changes in the distribution. Similarly, in sentiment analysis of reviews, product features

change over time, and even the word polarity may change in different time periods (Blitzer et al., 2007; Hamilton et al., 2016; Yin et al., 2018). In computer vision problems, e.g., object detection, domain shifts happen due to locations and pose changes (Hoffman et al., 2013). This distribution discrepancy between training data and test data gives rise to a need for domain adaptation approaches.

Domain adaptation is a type of transfer knowledge approach that can be used to address the lack of labeled data for a target domain by using labeled data from one or more related source domains. The domain shift between source and target data is generally addressed by using unlabeled data from the target domain. Thus, a domain adaptation approach aims to build a good model for a target domain through utilizing available labeled data from one or more related, but different, source domains, and also a large amount of unlabeled data from target domain. Just like labeling reviews for all types of products in sentiment analysis is prohibitive, for social media crisis data classification, labeling every type of disaster or crisis event is also prohibitive, especially when many variations of disasters or crisis event can happen given climate change and environmental challenges. Therefore, domain adaptation approaches that make use of labeled source data and target unlabeled data represent a promising direction for social media crisis data classifications as compared to traditional supervised machine learning approaches. However, domain adaption approaches have not been studied much in the context of emergency management. Thus, this research aims to contribute to this area and focuses on developing and applying domain adaptation approaches to classify and filter social media crisis data to help disaster management.

1.2 Social media crisis data classification

1.2.1 Value and challenges

Social media platforms have significantly changed the way people communicate, especially during emergencies. Due to social media's ubiquitous presence, rapid and easily accessible nature, people often go to social media to make sense of various events (Landwehr and Carley,

2014; Stefan et al., 2018), especially those in areas affected by a disaster. Therefore, social media data from platforms like Twitter are seen to have intrinsic value for both emergency response organizations and victims of disasters, due to their growing ubiquity, communications rapidity, and cross-platform accessibility (Vieweg et al., 2010). Governments that aim to improve situational awareness during emergencies are also starting to value such on-the-ground information offered by eyewitnesses or average citizens on social media during emergency events (Hom, 2014; Hughes et al., 2014; Palen et al., 2009; Reuter et al., 2015). For example, the Federal Emergency Management Agency (FEMA) launched an app as its own social media platform in June 2016, which not only lets users receive information but also lets them submit disaster-related images (Transportation Research Board and National Academies of Sciences, Engineering, and Medicine, 2017). In a workshop on the role of social media in disaster response back in 2012 (National Research Council, 2013)[1], different emergency departments shared their experience with social media. Los Angeles Fire Department (LAFD) was able to identify an explosion in the airport as a minor incident, by engaging the public through the location-based social media platform Foursquare, as well as Twitter. The National Earthquake Information Center (NEIC) also used Twitter to disseminate alerts and detect earthquakes through a tool watching for a sharp increase in the rate at which keywords associated with earthquakes were used. Although limited, this tool could extend coverage in areas with little instrumentation, and could also provide a backup should instruments fail or be offline. Centers for Disease Control and Prevention (CDC) used social media both to help detect emerging threats and to educate the public about how to respond.

According to Reuter et al. (2012), the usage of social media during emergency situations can be categorized according to four patterns. Considering citizens (C) and authorities (A), such as emergency services, these four patterns are: 1) A2A: communications between organizations of response teams; 2) C2C: communications between the affected public (i.e., citizens) and volunteers via social media; 3) C2A: citizens report or post information on social media, and response organizations monitor and analyze such citizen generated content;

[1]Convened by National Research Council (NRC) Computer Science and Telecommunications Board under request of Department of Homeland Security's (DHS's) Science and Technology Directorate

4) A2C: disseminating warnings, alerts and other information from the emergency organizations to the public. Another study about social media usage during the November 2015 Paris Terrorist Attacks showed that authorities gained situational awareness via social media (C2A), and that citizens used social media to help one another (C2C) [2].

However, there are still many challenges that prevent emergency organizations from extensively adopting social media in practice. Besides management barriers such as limited staff and budget challenges, lack of policies and guides, there are also many technical challenges, including emergency responders' limited knowledge regarding the collection of information, the obvious information overload, the reliability or trustworthiness of the information sources (National Research Council, 2013; Plotnick et al., 2015; Reuter et al., 2015), and also issues of quantification of performance, focus of attention, and translation of reported observations into a form that can be used to combine with other information (Mendoza et al., 2010; Starbird et al., 2010; Tapia et al., 2011, 2013).

Information can accumulate very quickly on social media during emergency situations. The information from eyewitnesses or victims of the disaster, can be easily buried deep under the torrent of news reports of the emergency situation, and also of other unrelated or even untrusted information. This makes it hard for the public to resort to the social media for help, and for the emergency organizations, such as volunteer organizations and first responders, to gain situational awareness through social media. Therefore, automated filtering tools are expected to be key in solving this problem (National Research Council, 2013). This is where machine learning and natural language processing (NLP) techniques can help.

In recent years, researchers have been optimistic about the potential value of social media data in helping emergency teams to improve situational awareness and emergency response, provided that accurate information can be automatically identified (Castillo, 2016; Meier, 2015; Palen and Anderson, 2016; Qadir et al., 2016; Reuter et al., 2015; Watson et al., 2017). Many researchers have worked on applying these techniques to separate relevant and irrelevant information, categorize sources and information types, to recognize rumors,

[2] https://whova.com/embedded/subsession/iscra_201805/348169/348170/

etc. (Castillo, 2016; Imran et al., 2015). However, to train a good model, many machine learning algorithms, especially the data-hungry deep learning methods, need large amounts of labeled data, which will not be available in a short time for a new event that is just happening. One solution for this problem is to use historical labeled data from previous events. But as each event is unique in terms of type, location, culture, people involved, etc., and different events may cause different social media responses (Palen and Anderson, 2016; Palen and Hughes, 2018), supervised classifiers trained on a previous emergency event may not generalize well on a current event in practice (Imran et al., 2013a, 2015). On the other hand, as information about the on-going event spreads, unlabeled data for the event can be quickly accumulated and extracted without much effort. Such unlabeled data can help to align the previous labeled data to the event of interest using domain adaptation approaches. Therefore, domain adaptation approaches could be more effective than supervised methods in building automated filtering tools for disaster management (Imran et al., 2015; Li et al., 2015, 2017a, 2018a).

1.2.2 Research goals

This research focuses on the analysis of user-generated data and aims to build automated filtering and classification tools for emergency responders and other humanitarian and disaster relief organizations (e.g., volunteer organizations), or even for citizens who want to know more about the events. Concretely, the user-generated data in this research is Twitter text data, i.e., tweets, but the approaches can be adapted to analyze short text from other social media platforms as well.

Social media is a general concept identifiable in many different platforms through the Internet, web pages like Wikipedia pages, video sharing platforms like YouTube, and also well-known social networks like Facebook, and micro-blogging platforms like Twitter, Instagram, etc. While both text (conversations), and images/videos can be useful for emergency responders, this research is focused on text classification of Twitter data, i.e., tweet classification. Reasons for focusing on classification of text in Twitter as opposed to other social

media platforms include: 1) Twitter is one of the most popular social media platforms world wide; 2) Twitter data is more readily available as compared to data from other social media platforms such as Facebook; 3) there are more prior works focused on Twitter data, therefore, more public datasets are available for this platform, which enable training and evaluation of approaches. Specifically, this research aims to addresses the following problems using domain adaptation approaches:

- Identify tweets relevant to an emergency event. Given a new emergency event, the first information filtering step is to separate relevant information from irrelevant information.

- Identify informative tweets among the relevant tweets. Given a new emergency event, some relevant tweets may not be informative to the responders, if they don't contain some specific information. Therefore, the relevant tweets need to be filtered further to select the informative ones.

The informative tweets can be further classified into different categories useful for different response teams, such as medical, infrastructure damage or donation. The ultimate goal for each of the above categorization problems is to build classifiers through training on the available labeled and unlabeled data using domain adaptation approaches and other NLP techniques, so that when there is a crisis, these classifiers can be deployed as filtering tools in practice.

1.3 Contributions and outline

The contributes of this thesis can be summarised as follows:

- We contribute to the crisis informatics researches by being the first to bring in domain adaptation approaches for social media crisis data classification.

- In the single-source setting, we propose iterative domain adaptation approaches based on Naive Bayes and Self-Training or EM strategies, which are simple but perform

better than supervised Naive Bayes classifiers, and can be potentially used in practice.

- Also in the single-source setting, we propose a hybrid domain adaptation approach which combines Correlation Alignment and Self-Training. Like the other proposed domain adaptation approaches, this approach is simple and computationally efficient, with potential for being used in practice.

- In the multi-source domain adaptation setting, we evaluate how different representations using word embeddings and sentence encoding models perform with different supervised learning models, including traditional models and deep neural network models, and provide insights for potential practical usage.

- We train crisis specific word embeddings with a crisis tweet corpus crawled locally. The resulting embeddings can be used for other crisis tweet classification tasks.

- We propose to build domain adaptation models on top of state-of-the-art pre-trained language models (BERT) for social media crisis data classification.

The outline of this thesis is as follows:

- Chapter 2: *Background and literature review*

  In this chapter, we present an overview of machine learning basics and some supervised learning algorithms used in this thesis. We define the concept of domain adaptation and provide a thorough literature review of domain adaptation approaches, as well as a literature review of related research on social media crisis tweets.

- Chapter 3: *Datasets*

  In this chapter, we introduce the datasets used in this thesis, including datasets of tweets labeled as relevant or non-relevant from several disasters, and similarly datasets of tweets labeled as informative or non-informative. We also describe the process we used to collect a large amount of tweets during the 2017 Hurricane season (which included Hurricane Irma, Hurricane Harvey and Hurricane Maris) and during the 2017

Mexico Earthquake. The collection of tweets crawled locally is used as unlabeled data to train crisis specific word embeddings.

- Chapter 4: *Self-labeling and Correlation Alignment approaches with Naive Bayes*

  In this chapter, we propose domain adaptation approaches based on Naive Bayes with Expectation Maximization (EM) and self-training strategy (these two strategies are also being referred as self-labeling methods). We also perform a hyper-parameter analysis to provide guidance for potential practical application. We further compare the self-training approach with a feature representation based approach, Correlation Alignment (CORAL), used on top of supervised Naive Bayes classifiers, and propose a hybrid model combining self-training and CORAL on top of Naive Bayes models. We experiment with different source and target disaster pairs on predicting the disaster related tweets.

  This chapter is based on four published works (Li et al., 2015, 2017a, 2018a,c),

- Chapter 5: *Domain adaptation using embedding-based approaches*

  To apply the approaches in the previous chapter, we need to build a classifier for each target disaster. Ideally, we would like to have a classifier that is ready-to-use whenever a disaster strikes. Therefore, in this chapter, we experiment with embedding-based approaches that use pre-existing embedding models to transfer knowledge. Concretely, we experiment with embeddings at word-level, sentence-level, and also pre-trained language model based contextual-level embeddings under the multi-source domain adaptation setting. We first experiment with simple adaptation approaches with pre-trained word embeddings and sentence encoding models. We evaluate how different representations using pre-trained word embeddings and sentence encoding models perform with supervised learning models. We evaluate pre-trained word embedding with deep learning models, including Neural Networks, Convolutional Neural Networks and Long Short-Term Memory networks. Then, we experiment with approaches based on the state-of-the-art pre-trained language models, specifically BERT, and apply a multi-

source domain adversarial model using the representations computed from pre-trained BERT model.

Word-embeddings are pre-trained on very large unlabeled corpora, and as a result they capture semantic meanings (e.g., similar words are close in the embedding space). Given such embeddings, the goal here is to investigate whether these pre-trained word embeddings or sentence encoding models can be used as simple representation to produce a domain adaptation approach for crisis tweet classification. We further apply the self-training strategy on top of the CNN models, and experimentally show that the resulting model performs better than the supervised models on crisis relevant versus irrelevant task. Pre-trained language models have shown to improve transfer learning for many NLP tasks. We build classifiers based on BERT and compare with an adversarial model using BERT representations.

This chapter is based on published work (Li et al., 2018b) and unpublished work (Li and Caragea, 2020a) (to be submitted).

- Chapter 6: *Conclusions and future directions*

  The last chapter summarizes the thesis and discusses ideas for future work.

Chapter 2

Background

In this chapter, we review the fundamentals of machine learning. We first overview the basics of machine learning in Section 2.1. Then, we present the supervised learning concepts and the supervised learning classifiers used in this thesis in Section 2.2, and after that we introduce unsupervised learning and semi-supervised concepts in Section 2.3 and 2.4, we will briefly discuss the learning algorithms that are related to domain adaptation. Finally, we will give the definition of domain adaptation in Section 2.5. This chapter will provide the necessary background for the following chapters.

2.1 Machine learning basics

One definition of **machine learning** that is widely quoted is as follows:

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" (Mitchell, 1997).

For example, for text classification, the task T can be classifying an email to be spam or not-spam, or in our crisis tweet classification case either relevant to a disaster or not, informative or not, and so on. Performance P can be accuracy which measures the percentage

of examples for which the model outputs the correct answer. The experience E would be the text with labels.

Depending on how the computer program is allowed to process (experience) the data, this program/algorithm can be broadly categorized as **supervised** learning or **unsupervised** learning (Goodfellow et al., 2016)[1]. In supervised learning, an example in the dataset is associated with a label provided by a teacher who shows the algorithm what to output given the example. In unsupervised learning, the learning program only sees the examples, no labels are provided, and the goal is to learn the useful properties of the structure of the data. One example is clustering, which is used in applications such as market segmentation where we want to separate a company's customers into different groups for different marketing strategies.

Building a machine learning algorithm

To design machine learning algorithms for learning tasks, we first need to represent the data in the format of input-output that the computer can understand. Typically, each example in the dataset (e.g. document or tweet) is taken as an input and is represented as a vector $\mathbf{x} \in \mathbb{R}^n$ that has n features, with feature values corresponding to an attribute (e.g. word) of that data example. A set of examples can be represented as a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ containing m examples with each example as a row. The observed example set is called the training set, and the unobserved example set called the test set. For supervised learning, an output (label) y_i (typically $y_i \in \mathbb{R}$) is also provided with every input $\mathbf{x}_i$, and correspondingly label vector $\mathbf{y}$ for an entire set of examples $\mathbf{X}$. From this point on, we denote the training dataset as D, which includes training examples $\mathbf{X}$ and their labels $\mathbf{y}$ in supervised learning and only $\mathbf{X}$ in unsupervised learning.

From a **probability perspective**[2], machine learning often involves probabilistic infer-

[1]Note that there is also reinforcement learning, in which there is a feedback loop between the algorithm and its experience as it interacts with the environment. We may see it in literature reviews being used in domain adaptation for selecting source labeled examples close to the target domain to improve performance, but we will not go into those details here.

[2]Key Ideas in Machine Learning, Tom M. Mitchell, http://www.cs.cmu.edu/%7Etom/mlbook/keyIdeas.pdf

ence of the learned model from the training data and prior probabilities. Let X denote the random variable associated with the training example[3], and let Y be the random variable associated with the label in supervised learning. The training data is usually viewed as drawn *independent and identically distributed* (i.i.d) from some unknown distribution $P(X, Y)$ in supervised learning or $P(X)$ in unsupervised learning. In theory, if we know the joint distribution we can solve any classification, regression, or other function approximation problem defined over these variables[4]. For example, in classification, or supervised learning in general, we are actually interested in the conditional distribution $P(Y|X)$, which can be derived from $P(X, Y)$ with *Bayes' rule*. But in machine learning we never know the true distribution. What we do know is the given data samples, from which we can get the empirical distribution $\hat{P}(D)$. We can estimate empirical probability distribution from samples with a model that has parameters $\boldsymbol{\theta}$, denoted as $P(D; \boldsymbol{\theta})$. There are two main probabilistic principles for choosing $\boldsymbol{\theta}$: *Maximum Likelihood Estimation* (MLE) and *Maximum a Posteriori Probability* (MAP) estimation.

MLE estimates one or more parameters $\boldsymbol{\theta}$ that define a probability distribution based on the principle that we choose the value of $\boldsymbol{\theta}$ that makes the observed data D most probable, i.e. find $\hat{\boldsymbol{\theta}}$ that maximizes the likelihood of the data $P(D; \boldsymbol{\theta})$:

$$\hat{\boldsymbol{\theta}}_{MLE} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ P(D; \boldsymbol{\theta}) \tag{2.1}$$

We can often solve this maximization problem in two steps. First, make an assumption about what type of distribution the data are sampled from, plug in the terms for the distribution and take the log of the function. Second, compute the derivative of the function and set it equal to zero to we get the optimal value for $\boldsymbol{\theta}$. Based on the first step, we can say that we

[3]We use uppercase letters to denote random variable, including both vector and non-vector variables. If X is a vector $X = \langle X_1, X_2..X_n \rangle$ we use X_i to refer to each random variable or ith feature in X. We use lower case symbols to refer to values of random variables, e.g. $X_i = x_{ij}$ refers to random variable X_i taking on its jth possible value. And we abbreviate by omitting variable names, for example abbreviating $P(X = \mathbf{x} \mid Y = y)$ to $P(\mathbf{x} \mid y)$, $P(X_i = x_{ij} \mid Y = y_k)$ to $P(x_{ij} \mid y_k)$.

[4]Tom M. Mitchell, Estimating probabilities, http://www.cs.cmu.edu/~tom/mlbook/Joint_MLE_MAP.pdf

are actually maximizing the *log likelihood*.

$$\hat{\boldsymbol{\theta}}_{MLE} = \underset{\boldsymbol{\theta}}{\text{argmax}} \log P(D; \boldsymbol{\theta}) \tag{2.2}$$

MLE can find the *true* parameters if the dataset size m is large and the distribution (model) assumption is correct. However, when m is small and the model is incorrect, MLE can fail.

In MAP estimation, we choose the parameters of $\boldsymbol{\theta}$ that are most probable given the observed data D and our prior knowledge or assumptions of $\boldsymbol{\theta}$, summarized by $P(\boldsymbol{\theta})$, i.e. find $\hat{\boldsymbol{\theta}}$ that maximizes the posterior $P(\boldsymbol{\theta} \mid D)$:

$$\hat{\boldsymbol{\theta}}_{MAP} = \underset{\boldsymbol{\theta}}{\text{argmax}}\ P(\boldsymbol{\theta} \mid D) = \underset{\boldsymbol{\theta}}{\text{argmax}}\ P(D \mid \boldsymbol{\theta}) P(\boldsymbol{\theta}) \tag{2.3}$$

The difference from MLE is the extra $P(\boldsymbol{\theta})$; here $\boldsymbol{\theta}$ is not a parameter anymore, it's a random variable. If we think MLE corresponds to the intuitive notion that we should base probability estimate on observed ratios like a set of coin tosses, then MAP corresponds to the the intuition that we can represent prior assumptions of $\boldsymbol{\theta}$ by making up "imaginary" tosses to reflect these assumptions. Like in a set of coin tosses where we only see heads, based on our prior knowledge that there are two sides, we imagine some tails so that the probability that a tail appears is not zero. Many algorithms are derived based on these two principles, and we will see MLE being used to derive cost functions in some machine learning algorithms.

From an **optimization perspective**, machine learning tasks can be formulated as optimization problems to discover the best parameters that optimize a cost function (an objective function) given the dataset. The algorithms can usually be described as a combination of a specification of dataset D, a model or hypothesis function $h()$ which maps the input to the output, a *cost function* $J(\boldsymbol{\theta})$ (or *objective function* or *loss function*) and an optimization procedure (Goodfellow et al., 2016). Many learning algorithms, especially deep learning models, can be described in this way. This recipe supports both supervised and unsupervised learning. In supervised learning the cost function is constructed with respect to the output y,

while in unsupervised learning the cost function is constructed only relative to the input examples $\mathbf{x}$ like in dimension reduction. In both settings, a cost function can be derived based on the MLE principle. One common cost function in supervised algorithms, such as *logistic regression* or *neural networks*, is *cross-entropy loss*. This cost function corresponds to the negative log-likelihood (NLL) of the model parameters; therefore minimizing cross-entropy loss is equivalent to maximizing the likelihood of the model parameters. One common optimization procedure is gradient descent, where we update the model's parameters $\boldsymbol{\theta}$ in the opposite direction of the gradients $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ with a learning rate α.

These two perspectives of machine learning are compatible. In many cases, deriving a learning algorithm based on MLE or MAP principle involves first defining an objective function in terms of parameters of the hypotheses and training data, and then applying an optimization procedure to find the optimal parameter values. We will see in a later overview section that in some cases, the learned model may itself contain explicit probabilities, for example the learned parameters in a Naive Bayes classifier. In other cases, even though the model parameters do not correspond to specific probabilities, for example in neural network models, we may still find it useful to view the algorithm as performing probabilistic inference to find the maximum likelihood or maximum a posterior probability network parameters' values[5].

Generalization and regularization

The main challenge for machine learning is the ability to perform well on previously unseen inputs, which is called *generalization*. The unseen inputs are referred to as test data, which are assumed to be drawn from the same distribution as the observed training data. The error that the machine learning model makes on the observed training data is called *training error*, and the error on test data is called *test error* or *generalization error*. In practice, we usually draw a portion of the training data and treat it as unseen data, which we call the *validation set*. The test error thus is the error on the validation set. An ideal model will achieve low training error as well as low test error close to the *Bayes error*, which is the error

[5]Key Ideas in Machine Learning, Tom M. Mitchell, http://www.cs.cmu.edu/%7Etom/mlbook/keyIdeas.pdf

incurred by an oracle knowing the true distribution of the data. The ability to

1. make the training error small;

2. make the gap between training and test error small;

determines how well a machine learning algorithm performs. If the training error is high, the model is *underfitting* the data, or the model has high *bias* from a probabilistic perspective. In this case, we should increase the *capacity* of the model, which is its ability to fit a wide variety of functions. If the gap between training error and test error is large, then the model is *overfitting* data, or the model has high *variance*. In this case, we want to decrease the capacity of the model. When the test data distribution is different from training data, such as in domain adaptation problems, the test error is much more important than training error because we are mainly interested in the test (target) data.

Regularization is another way to change the capacity of the model, and is usually used to prevent overfitting. By adding a regularization term to the cost function, we can change the model to prefer certain functions while searching in the hypothesis space:

$$\tilde{J}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \alpha\Omega(\boldsymbol{\theta}) \tag{2.4}$$

The most commonly used regularization forms are l_1 and l_2 norm, where $\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1$ and $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|_2^2$ respectively. Intuitively, both terms encourage the model to find parameters that are sparse where weights for some features will be zero. Therefore, these terms can help prevent overfitting. While l_1 results in a solution more sparse than l_2 in general, derivatives of l_2 are easier to compute. The prior knowledge $P(\boldsymbol{\theta})$ in MAP estimation corresponds to the regularization terms in some algorithms.

2.2 Supervised learning

Now that we have introduced the concept of supervised learning, we can further formalize that concept. In supervised learning, the training data comes in input-output pairs and is

denoted as $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \subseteq \mathcal{X} \times \mathcal{Y}$ where $\mathbf{x}_i \in \mathcal{R}^n$ is the input instance, y_i is its label, $\mathcal{X} = \mathcal{R}^n$ is the feature space or input space and $\mathcal{Y}$ is the label space. If $\mathcal{Y} = \{0, 1\}$ or $\{1, -1\}$, then the machine learning task is called a binary classification. If $\mathcal{Y} = \{1, 2 \ldots C\}$ where $(C \geq 2)$, then it is a multi-class classification, and if $\mathcal{Y} = \mathbb{R}$, then it is a regression task.

The goal of supervised learning is to learn a function h such that $h(\mathbf{x}) \approx y$ or $h(\mathbf{x}) = y$ with a high probability. From a probability perspective, $h(\mathbf{x})$ can be viewed as probability $P(y|\mathbf{x})$. We can estimate this conditional probability through estimating $P(\mathbf{x}, y)$ as we have discussed. These kinds of algorithms are usually referred to as *generative models*. We can also model it directly with $P(y|\mathbf{x}; \boldsymbol{\theta})$ and find the parameters that maximize the conditional likelihood. These models are referred to as *discriminative models*. The Naive Bayes model is a typical example of a generative model, and logistic regression is usually referred to as the discriminative counterpart of Naive Bayes. There are other traditional supervised learning algorithms that are nonprobabilistic and also used in this thesis, such as Support Vector Machines (SVM), K-nearest Neighbors and Random Forests. We will give the overviews of these models along with neural networks and some deep learning models.

2.2.1 Traditional supervised learning algorithms

Naive Bayes

As a generative model, Naive Bayes models $P(y|\mathbf{x})$ through estimating $P(\mathbf{x}, y)$, according to Bayes' rule:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}, y)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|y)P(y)}{\sum_{y'} P(\mathbf{x} \mid y')P(y')} \tag{2.5}$$

So we can estimate $P(y)$ and $P(\mathbf{x}|y)$ instead. Estimating $P(y)$ is simple. For example, if Y takes on discrete binary values, estimating $P(Y)$ reduces to coin tossing. With maximum likelihood estimation, we simply need to count how many times we observe each outcome

(in this case each class):

$$P(y = c) = \frac{\sum_{i=1}^{m} I(y_i = c)}{m} = \hat{\pi}_c \tag{2.6}$$

Estimating $P(\mathbf{x}|y)$ however is more complicated, as it involves n features and therefore joint distribution of n random variables, which can be very complex. An additional assumption that Naive Bayes makes is that the features are conditional independent given the label. That means we can estimate:

$$P(\mathbf{x}|y) = \prod_{j=1}^{n} P(x_j|y) \tag{2.7}$$

where x_j is the value for feature j. Then we can derive our classifier as:

$$\begin{aligned} h(\mathbf{x}) &= \operatorname*{argmax}_y P(y|\mathbf{x}) \\ &= \operatorname*{argmax}_y \frac{P(\mathbf{x}|y)P(y)}{\sum_{y'} P(\mathbf{x} \mid y')P(y')} \\ &= \operatorname*{argmax}_y P(\mathbf{x}|y)P(y) \\ &= \operatorname*{argmax}_y \prod_{j=1}^{n} P(x_j|y)P(y) \\ &= \operatorname*{argmax}_y \log(P(y)) + \sum_{j=1}^{n} \log(P(x_j|y)) \end{aligned} \tag{2.8}$$

Thus the estimation simplifies to each feature. Depending on the types of feature values, there are usually three common cases. Features can be categorical or just binary, or multinomial or continuous. We can use multivariate Bernoulli Naive Bayes to model binary features. Multinomial Naive Bayes models multinomial features where a feature value is a count, such as the count of how many times a word appears in one document. Gaussian Naive Bayes models continuous features. In our experiments, for short text samples such as tweets, Bernoulli Naive Bayes model usually works better than a multinomial model. Therefore the base classifier used in Chapter 5 is the Bernoulli Naive Bayes model. When there are continuous features, Gaussian Naive Bayes is used, which can be shown to be exactly the

same as logistic regression under some assumptions[6].

Logistic regression

For binary classification, logistic regression models assume that $P(y|\mathbf{x})$ takes the form:

$$P(y = 1|\mathbf{x}_i; \mathbf{w}) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x}_i)}} \tag{2.9}$$

And

$$P(y = 0|\mathbf{x}_i; \mathbf{w}) = 1 - P(y = 1|\mathbf{x}_i; \mathbf{w}) \tag{2.10}$$

where $\mathbf{w} \in \mathcal{R}^n$ is the parameter vector that has the same dimensions as the input $\mathbf{x}$. It corresponds to the probabilistic parameters $\boldsymbol{\theta}$ in the previous discussion. We can always add an offset b into $\mathbf{w}$ through an additional constant dimension to $\mathbf{x}_i$ to form an affine function, but we will absorb b for simplicity and just use a linear function in place of an affine function.

We can think of logistic regression as linear regression generalized to a classification task, where we use a *sigmoid function* or *logistic function* $g(z) = \frac{1}{1+e^{-z}}$ to squash the real-valued output into a value between 0 and 1. When $z = 0$, $g(z) = 0.5$, and $g(z)$ tends to 0 for negative infinity and to 1 for positive infinity. So if $\mathbf{w}^\top \mathbf{x}_i >> 0$ then the probability that $y = 1$ is close to 1. Therefore we interpret the output value as a probability $\hat{y} = P(y = 1|\mathbf{x}_i; \mathbf{w}) = g(\mathbf{w}^\top \mathbf{x}_i)$. Notice we can rewrite equation 2.9 and 2.10 as follows:

$$P(y|\mathbf{x}; \mathbf{w}) = (\hat{y})^y (1 - \hat{y})^{1-y} \tag{2.11}$$

We can then estimate the parameters using MLE given that the data points are drawn i.i.d.

$$\begin{aligned} \hat{\mathbf{w}}_{MLE} &= \underset{\mathbf{w}}{\text{argmax}} \prod_{i=1}^{m} P(y_i \mid \mathbf{x}_i; \mathbf{w}) \\ &= \underset{\mathbf{w}}{\text{argmin}} \frac{1}{m} \sum_{i=1}^{m} -y \log(\hat{y}) - (1 - y_i) \log(1 - \hat{y}_i)) \end{aligned} \tag{2.12}$$

[6]Refer to lecture notes, https://alliance.seas.upenn.edu/ cis520/dynamic/2020/wiki/index.php?n=Lectures.Logistic

The last part is the averaged cross-entropy loss over all data points, which we usually refer to as the cost function of logistic regression. We can use gradient descent to find the optimal $\mathbf{w}$ that minimizes the cost function. We then get our classifier as demonstrated in Equation 2.9 to predict the probability of its label given a test sample.

For multi-class classification, we learn a separate set of weights $\mathbf{w}_c$ for each class $y = c$. We then use a *softmax* function to squash the values to obtain a categorical distribution:

$$P(y = c|\mathbf{x}; \mathbf{w}) = \frac{e^{\mathbf{w}_c{}^\top \mathbf{x}}}{\sum_{c'=1}^{C} e^{\mathbf{w}_{c'}^\top \mathbf{x}}} \tag{2.13}$$

where we sum over the scores for all C classes to normalize the scores to a distribution.

Support Vector Machines

Support Vector Machines (SVM) algorithm (Boser et al., 1992; Cortes and Vapnik, 1995) is one of the most influential supervised learning algorithms. Similar to logistic regression, SVM is also driven by a linear function; unlike logistic regression, it does not provide probabilities but only outputs a class sign. For a binary classification task, $y \in \{-1, 1\}$, SVM defines a linear classifier $h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$. If the data D is linearly separable by a hyperplane, then SVM aims to find the one hyperplane that maximizes the distance to the closest data points from both classes, i.e. the hyperplane with the *maximum margin*. A hyperplane is defined through $\mathbf{w}$, b as a set of points such that $\mathcal{H} = \{\mathbf{x}|\mathbf{w}^T\mathbf{x} + b = 0\}$. Let the margin γ be defined as the distance from the hyperplane to the closest point across both classes. These closest points are also referred to as **support vectors**. The objective is to maximize the margin γ under the constraints that all data points must lie on the correct side of the hyperplane, which is eventually transferred to the optimization problem:

$$\begin{aligned} &\min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2 \\ &\text{s.t.} \quad \forall i \; y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \end{aligned} \tag{2.14}$$

This is a problem with a convex quadratic objective and linear constrains, which can be solved by quadratic programming (QP) software. One important result we get after optimization for $\mathbf{w}$ is that

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \tag{2.15}$$

which is a linear combination of samples. Furthermore, only α_i of the support vectors are non-zeros. So now given sample $\mathbf{x}$, its decision function is decided by its dot product with the training examples.

$$\mathbf{w}^\top \mathbf{x} + b = b + \sum_{i=1}^{m} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) \tag{2.16}$$

This enables us to apply the *kernel* trick with SVM, which is useful for machine learning algorithms involving the dot products between examples. The kernel trick allows us to replace $\mathbf{x}$ with output of $\phi(\mathbf{x})$, a feature function. The dot product with a function $k(\mathbf{x}, \mathbf{x}_i) = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i)$ is called a *kernel*. If the data samples are not linearly separable, we can use kernel to map them to a higher dimension space where they may be linearly separable. This kernel trick is powerful for two reasons. First, it enables us to learn models that are nonlinear of $\mathbf{x}$ but still use convex optimization which is guaranteed to converge efficiently. Second, kernel function k often is significantly more computationally efficient than naively constructing the dot product of two $\phi(\mathbf{x})$ vectors. One commonly used non-linear kernel in SVM is the radial basis function (RBF) or Gaussian kernel, $k(\mathbf{x}, \mathbf{z}) = e^{-\frac{(\mathbf{x}-\mathbf{z})^2}{\sigma^2}}$.

SVM can be generalized to multi-class classification using a one vs all strategy where we train one classifier for each class in $\{1, 2, \ldots, C\}$ to distinguish it from the rest.

K-Nearest Neighbors and Random Forests

K-Nearest Neighbors (KNN) is a simple non-probabilistic and non-parametric learning algorithm. Parameters of non-parametric algorithms depend on the number of training examples. KNN assumes that similar inputs should have similar outputs, with similarity being measured by a distant metric. For a test input $\mathbf{x}$, KNN assigns the most common label amongst its k most similar training inputs. Once we select the distant metric and k,

we can predict the label for an input. The most common distance metric is the *Minkowski distance*:

$$\text{dist}(\mathbf{x}, \mathbf{x}') = \left(\sum_{j=1}^{n} |x_j - x'_j|^p \right)^{1/p} \tag{2.17}$$

when $p = 1$, it is the *Manhattan distance*, when $p = 2$, it is the *Euclidean distance*. The value of k can be from 1 to the size of the training set, with computation cost increasing with k. If k equals the training data size m, KNN can achieve minimum training error, however the computation cost becomes extremely high. If there is infinite training data, a test point will have infinitely many training set neighbors at distance zero. If we allow all of these neighbors to vote, the procedure converges to the Bayes error rate. This means that KNN has high capacity that enables it to obtain high accuracy given a large training set but it may generalize badly if the training set is small. Another weakness of KNN is that it cannot learn whether one feature is more discriminative than others. KNN also suffers from the curse of dimensionality, as the distance between two points increase drastically from low dimension spaces to high dimension spaces. When the dimension n approaches infinity, and the points drawn from a probability distribution stop being similar to each other, the KNN assumption breaks down.

Random Forests (Breiman, 2001) is a *bagging* (bootstrap aggregation) (Breiman, 1996) algorithm based on *decision trees*. For decision trees, we build a tree structure using the training data. Each node in the tree divides the space into regions with similar labels. The root node represents the whole training set, which is split to the left and right child nodes based on one feature or dimension with a value threshold. The leaf nodes are again split until eventually all data points of one leaf are in the same class or cannot be split any further (in the rare case with two identical points of different labels). In bagging, we first sample d different subsets of D (with replacement) and then apply a learning algorithm to each subset to learn a model. The final model is the average of each subset model. The Random Forests algorithm is essentially bagging applied to a Classification And Regression Tree (CART) algorithm (Breiman et al., 1984) with full depth (max-depth=∞), where at each split only $k \leq n$ randomly sampled features are evaluated to find the best splitting point.

The construction of a single tree is independent from earlier trees, thus making Random Forests an inherently parallel algorithm (Mohan et al., 2011). There are two parameters that need to be tuned for Random Forests: the number of trees in the forest d and the number of features k that each node considers to find the best split. One commonly used value for k is $\sqrt{n}$. One advantage of Random Forests is that because of bagging, it doesn't overfit with increasing the number of trees. And another advantage is that it doesn't require much preprocessing. For instance, the features can have different scales.

2.2.2 Neural networks

Neural networks

In neural networks, the final hypothesis function can be viewed as a composition of several different functions. Recall our discussion about squashing linear regression output to the interval [0, 1] to derive logistic regression for classification. The function of logistic regression $h(\mathbf{x})$ is actually formed by chaining two functions together: a linear functions f and then a sigmoid function g with $h(\mathbf{x}) = g(f(\mathbf{x})) = g(\mathbf{w}\mathbf{x} + b)$ where $\mathbf{x} \in \mathcal{R}^n$ and $\mathbf{w} \in \mathcal{R}^{1\times n}$ [7]. Logistic regression can be seen as a simple neural network with just one layer and a sigmoid function as an *activation function*.

In neural networks, the information flows through different functions with each one taking the previous one's output as input. For instance, it can be in this form: $f(\mathbf{x}) = f^{(1)}(f^{(2)}(f^{(3)}(\mathbf{x})))$, where each function corresponds to a *layer*. In this example, $f^{(1)}$ is defined as the *first layer* and so on, and the overall length of the chain defines the *depth* of the model (Goodfellow et al., 2016). The input is referred to as the *input layer*, the last layer is called the *output layer* and the middle layers are *hidden layers*. Each layer performs a linear transformation on the input, which is then followed by a non-linear *activation function*. The activation function for an output layer of a classification task is usually sigmoid or softmax function. Activation function for each hidden layer can be different, though one common

[7] We change the parameter vector $\mathbf{w}$ from a column vector to a row vector in order to simplify later parameters' matrix symbol. We also put b explicitly to be consistent to conventions to neural networks.

hidden layer activation function is *rectified linear unit*(ReLU):

$$g(\mathbf{z}) = \max(0, \mathbf{z}) \tag{2.18}$$

Another common hidden layer activation function is the *hyperbolic tangent* or *tanh* function:

$$g(\mathbf{z}) = \frac{e^{\mathbf{z}} - e^{-\mathbf{z}}}{e^{\mathbf{z}} + e^{-\mathbf{z}}} \tag{2.19}$$

Each hidden layer can have many *units* with each unit acting similarly to a biological neuron, or loosely speaking, performing a logistic regression calculation if we use a sigmoid function. Therefore, we will have a parameter matrix $\mathbf{W}$ for each layer. The dimensions of the parameter matrix $\mathbf{W}$ for each layer is determined by the input and output layer units. One example of two hidden layers neural network for multi-class classification can be as follows:

$$\begin{aligned} \mathbf{h}_1 &= g_1(\mathbf{W}_1\mathbf{x} + b_1) \\ \mathbf{h}_2 &= g_2(\mathbf{W}_2\mathbf{h}_1 + b_2) \\ \hat{\mathbf{y}} &= \text{softmax}(\mathbf{h}_2) \end{aligned} \tag{2.20}$$

With the model, we can define a cost function and use gradient descent to find the optimal or local optimal values of $\mathbf{W}$s. The calculation process from input to output in a neural network is called *forward propagation*. The gradient descent process, which can be seen as allowing information flowing from cost function backward through the networks, is performed with a *back-propagation* algorithm.

Word embeddings

In many neural network models for text classification, especially those with deep neural network architectures, each word in the vocabulary V is usually represented as a vector. The vector $\mathbf{x}_t$ for word w_t is called the word embedding of w_t. Then, given a piece of text, we can view it as a sequence of words $w_1, \ldots, w_T$, which can be represented as a sequence of

the word vectors $\mathbf{x}_1, \ldots, \mathbf{x}_T$. This can then be the input to a neural network model, including a deep neural network model that we will introduce next. The simplest word embedding is one-hot embedding. If the vocabulary size is $|V|$, one-hot embedding maps each word in the vocabulary to a vector that has 1 at the position where the word appears in the vocabulary and $|V| - 1$ 0s. As pre-trained word embeddings such as Word2Vec (Mikolov et al., 2013a) work better with the deep learning models, pre-trained word embeddings are widely used.

Recurrent Neural Networks

Recurrent Neural Networks or RNNs (Elman, 1990; Hopfield, 1982; Jordan, 1986) are a family of neural networks for processing sequential data. Therefore, RNNs have been used not only in NLP tasks such as speech recognition, machine translation, text generation and text classification, but also in other areas such as biological sequence tasks. Depending on the task, RNNs can have different output formats. For example, in classification we just have a single output, a label or a probability distribution over labels. However, in machine translation, we have a sequence as the output. In text generation, RNNs are used to train a language model where the model predicts the next word given the previous sequence of words. Given a start word or token, the model will generate a sequence of words until it generates an end token.

In general, an RNN can be seen as a feedforward neural network that has a dynamic number of hidden layers depending on the input sequence. Given a sequence of inputs at each time step, it not only takes the input at this time step but also the output from the previous time step. The parameter matrix as well as the activation function can be shared for all time steps. One example for a task that has an output at each time step can be as follows: suppose the input sequence is $\mathbf{x}_1, \ldots, \mathbf{x}_T$ and we use a hyperbolic tangent activation function for hidden layers and a softmax function for each time step output. Then for each

time step from $t = 1$ to $t = T$, the forward propagation computes the following equations:

$$\begin{aligned} \mathbf{h}_t &= \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}_h) \\ \hat{\mathbf{y}}_t &= \text{softmax}(\mathbf{V}\mathbf{h}_t + \mathbf{b}_y) \end{aligned} \tag{2.21}$$

where the parameters are the bias vectors $\mathbf{b}_h$ and $\mathbf{b}_y$ along with weight matrices $\mathbf{W}, \mathbf{U}, \mathbf{V}$ for input-hidden, hidden-hidden, hidden-output connections, respectively.

Long-short term memory

One significant problem for RNNs is that during back-propagation, the gradients are multiplied with the same values over and over and tend to either vanish (become very small) or explode (become very large). Even if we assume that the model is stable (gradients not exploding), difficulties with long-term dependencies arise from the exponentially smaller weights given to long-term interactions compared to short-term ones. Long-short term memory (Gers et al., 2000; Hochreiter and Schmidhuber, 1997) networks (LSTM) are sometimes used instead of RNNs to retrain information for longer time spans. Intuitively, LSTM introduces mechanisms to choose what should be memorized or forgotten. A LSTM unit adds different gates to a RNN unit: a forget gate $\mathbf{f}_t$, an update gate $\mathbf{u}_t$ and an output gate $\mathbf{o}_t$, which are all functions of current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. Meanwhile, it also maintain a memory cell state $\mathbf{c}_t$, which is determined by the previous memory state $\mathbf{c}_{t-1}$ and forget gate $\mathbf{f}_t$ as well as the candidate current memory state $\tilde{\mathbf{c}}_t$ and the update gate $\mathbf{u}_t$. Take the same example as in RNNs, the LSTM will compute the following equations for

each time step:

$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_f) \\
\mathbf{u}_t &= \sigma(\mathbf{W}_u[\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_u) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o[\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_o) \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c[\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{u}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}
\tag{2.22}
$$

where σ is the sigmoid function, $\mathbf{W}$s and $\mathbf{b}$s are the parameter matrices, $\odot$ means element-wise product and $[\mathbf{x}_t, \mathbf{h}_{t-1}]$ means concatenation of the two to simplify the equations. Intuitively, the current memory state $\mathbf{c}_t$ is the combination of how much the model chooses to forget given the previous state, and how much to update given the current input. The final output of this time step is determined by both the output gate and the current memory state.

Both RNNs and LSTM can be extended to be bidirectional where the model is given the whole sequence, and runs forward first and then backward over the sequence. We can also stack multiple layers of the units to get a deeper model.

Convolutional Neural Networks

A Convolutional Neural Network (CNN) (Lecun et al., 1998) is another type of neural network used specifically for processing data with grid-like topology, such as images. In fact, CNNs are one of the dominant models in current computer vision research. CNNs are also used in NLP, specifically text classifications. We will next introduce the commonly followed CNN architecture for text classification (Kim, 2014).

Given a sequence of words and $\mathbf{x}_i \in \mathcal{R}^d$ as the word embedding for the i-th word, the whole sequence then is represented as $\mathbf{x}_{1:T} = [\mathbf{x}_1^\top; \ldots; \mathbf{x}_T^\top] \in \mathcal{R}^{T \times d}$. Let $\mathbf{x}_{i:i+j-1} \in \mathcal{R}^{j \times d}$ refer to the concatenation of j words $\mathbf{x}_i^\top, \mathbf{x}_{i+1}^\top, \ldots, \mathbf{x}_{i+j-1}^\top$. A convolution operation involves

a *filter* represented by a weight matrix $\mathbf{W} \in \mathcal{R}^{j \times d}$, which is applied to $\mathbf{x}_{i:i+j-1}$ to generate a feature c_i by

$$c_i = g(\mathbf{W} \cdot \mathbf{x}_{i:i+j-1} + b) \tag{2.23}$$

where $b \in \mathcal{R}$ is a bias term and g is the activation function. Sliding the filter over each possible window of j words of the sequence will produce a feature map $\mathbf{c} \in \mathcal{R}^{T-j+1}$

$$\mathbf{c} = [c_1, c_2, \ldots, c_{T-j+1}]^\top \tag{2.24}$$

where each entry can corresponds to a n-gram feature in traditional NLP which considers the j consecutive words as one feature. Then we can further apply *max-pooling* to get the one most important entry as the final feature of this filter:

$$\tilde{c} = \max(\mathbf{c}) \tag{2.25}$$

We can have hundreds of such filters with window size 2 and hundreds of filters with window size 3 and so on in CNNs, and the max-pooling from all feature maps are finally concatenated to a vector $\tilde{\mathbf{c}} \in \mathcal{R}^k$ where k is the number of filters. This vector can then be passed to the next layer or to the output layer for classification. If we have 100 filters with window size 2 and another 100 filters with window size 3, intuitively that corresponds to 100 bigram features and 100 trigram features for the given piece of text. This enables CNNs to capture local features with different spans.

We have discussed a lot on supervised learning, next we will move on to unsupervised learning, semi-supervised learning and finally domain adaptation.

2.3 Unsupervised learning

In unsupervised learning, we have a training set $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \subseteq \mathcal{X}$, where $\mathbf{x}_i \in \mathcal{R}^n$ is the input instance which is assumed to be drawn i.i.d. from a distribution, but the labels are not known. The goal is to learn the internal structure of the data or distribution. Some

common unsupervised tasks are: 1) *dimensionality reduction*, where we want to project the inputs to a lower dimensional space for purposes such as saving storage space, visualization or for more meaningful distance calculations. One common algorithm is *Principal Component Analysis* (PCA); 2) *clustering*, where we want to group similar inputs into clusters. *K-Means clustering*, a simple iterative clustering algorithm, is one common algorithm for this task. Given the dataset D, we first randomly pick k cluster centers among all data points. We then assign each data point to its nearest center. Finally, we update the center to be the mean of its assigned points. The last two steps are alternated until convergence; 3) *density estimation*, where we want to estimate the distribution of the data. This is used for applications such as anomaly detection.

Another type of unsupervised learning is focused on *latent representation learning*, which can been seen as one type of dimensionality reduction task. However its main purpose here is to learn a better and more compact representation of the data. One example is an autoencoder, which is a type of neural network that aims to output its input using an encoding and decoding process. Through the hidden layers, we can discover interesting structures in the data and use the hidden layer representations as latent representations of the original data.

Another type of unsupervised learning is sometimes called *self-supervised learning*, where we create heuristic labels from the unlabeled data and then turn the unsupervised problem into a supervised problem. Pre-trained word embeddings and language models can be seen as examples of self-supervised learning. We will leave this to later discussion.

Mixture of Gaussian and EM

One common model for density estimation is a mixture model, such as a *mixture of Gaussians*, where the assumption is that data is generated from a linear combination of K Gaussian distributions, $p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \Sigma_k)$. We can introduce a hidden variable $z \in \{1, \ldots, K\}$ for each observed sample $\mathbf{x}$, such that z indicates from which mixture component the sample originated, and write the marginal probability as $p(\mathbf{x}) = \sum_{k=1}^{K} P(z =$

$k)p(\mathbf{x} \mid z = k) = \sum_z P(z)p(\mathbf{x}, z)$. The parameters (π_k, μ_k, Σ_k) for $k = 1, \ldots, K$ in a Gaussian mixture model are estimated with an algorithm called *Expectation Maximization* (EM) (Dempster et al., 1977). EM finds the maximum likelihood solution for models with latent variables by estimating parameters in an iterative way. Intuitively, the idea of EM is much like K-Means. EM also alternates between two steps, E step and M step, until convergence. In the E step we estimate distributions of hidden variables given the model, and in the M step we estimate the model given the hidden variable estimates in E step. We can think of the hidden variables in K-Means as the cluster assignments. In K-Means, we assign a data point to just one cluster, which is a hard assignment. In EM, the cluster assignments for Gaussian mixtures are soft, meaning we assign a point to each cluster with a probability. EM is used in this thesis together with a base classifier for domain adaptation.

Autoencoder

As a type of neural network that copies its inputs to outputs, an autoencoder has two components: an encoder and a decoder. Let's just take one hidden layer architecture as an example. The encoder first maps the input to a vector such as:

$$\mathbf{h} = g(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.26}$$

where g is the activation function. Then the decoder reconstructs the input from the latent representation as:

$$\tilde{\mathbf{x}} = \tilde{g}(\tilde{\mathbf{W}}\mathbf{h} + \tilde{\mathbf{b}}) \tag{2.27}$$

Then the model is trained to minimize the reconstruction loss between the original and reconstructed input as the mean squared loss. In practice, deep neural network architectures such as RNNs or LSTMs are often used in autoencoders for text. Autoencoders are used in domain adaptation to learn latent domain representations.

2.4 Semi-supervised learning

Semi-supervised learning lies between supervised and unsupervised learning. In semi-supervised learning, we have some labeled data, $D_L = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{m'}, y_{m'})\} \subseteq \mathcal{X} \times \mathcal{Y}$, but usually not enough to learn a good supervised model. Thus, we also use a set of unlabeled data $D_U = \{\mathbf{x}_{m'+1}, \ldots, \mathbf{x}_m\} \subseteq \mathcal{X}$ that is drawn from the same distribution to help the supervised tasks. Here, the size of unlabeled dataset is usually much larger than labeled dataset. Since unlabeled data comes from the same distribution as the labeled data, unlabeled data can be used for more accurate probability distribution estimation and also to learn better representations. There are different types of semi-supervised learning algorithms, such as generative models with EM, and low density separation like Transductive SVM. Semi-supervised learning and domain adaptation both use unlabeled data, so we will see some algorithms used in semi-supervised learning also used in domain adaptation such as self-training (Yarowsky, 1995) and co-training (Blum and Mitchell, 1998; Chen et al., 2011a; Nigam and Ghani, 2000). We will discuss these methods in the reviews of domain adaptation approaches.

2.5 Domain adaptation

We can limit the labeled data even more strictly than in semi-supervised learning. For example, if we don't have any labeled data available, but we have labeled data from a related domain for the same learning task, then we can use domain adaptation approaches to perform the learning task. In domain adaptation problems, the domain where labeled data is available is referred to as the *source* domain, and the domain of interest where labeled data is either not available or available in a very small amount is referred to as the *target* domain. Domain adaptation is often mentioned together with transfer learning, and according to a seminal survey paper by Pan and Yang (2010), domain adaptation can be seen as a special case of transfer learning. We will follow the notation in this paper to give the definition of transfer learning and then domain adaptation.

Transfer learning

According to Pan and Yang (2010), transfer learning involves the concepts of a domain and a task. A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$ over the feature space, so we can denote a domain as $\mathcal{D} = \{\mathcal{X}, P(X)\}$. Recall from our discussion in previous sections, X is the random variable associated with training examples, and we can have a random vector $X = \langle X_1, X_2..X_n \rangle$ where X_i refers to each random variable or ith feature in X. Given a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task $\mathcal{T}$ consists of a label space $\mathcal{Y}$ and a hypothesis function $h(\cdot)$, which is the conditional probability distribution $P(Y \mid X)$ from a probability perspective that needs to be learned from the training data consisting of pairs of examples and labels. We can denote a task as $\mathcal{T} = \{\mathcal{Y}, P(Y \mid X)\}$[8].

Given a source domain $\mathcal{D}_S$, its task $\mathcal{T}_S$, and a target domain $\mathcal{D}_T$ and its task $\mathcal{T}_T$, the goal of transfer learning is to learn the target conditional probability distribution $P_T(Y_T \mid X_T)$ with knowledge and experience from $\mathcal{D}_S$ and $\mathcal{T}_S$ where either $\mathcal{D}_T \neq \mathcal{D}_S$ or $\mathcal{T}_T \neq \mathcal{T}_S$.

Let D_S denote the source labeled data, D_{SU} denote source unlabeled data, similarly D_T be the target labeled data and D_{TU} be the target unlabeled data. The case in which the target task is different from the source $\mathcal{T}_T \neq \mathcal{T}_S$ and D_T is available (usually in a small amount) is called *inductive transfer learning*. Here, if D_S is used, it is usually a multi-task learning case. If D_{SU} is used, it is sometimes referred to as *self-taught* learning (Raina et al., 2007) or *sequential transfer learning* in Ruder (2019) since unlabeled data is usually first used for representation learning, then the representation is sequentially used in the target task. Pre-trained models in our later discussion can fall into this category.

The case when $\mathcal{T}_T = \mathcal{T}_S$ but $\mathcal{D}_T \neq \mathcal{D}_S$, and only D_{TU} is available, is called *transductive transfer learning*. Here, domain differences $\mathcal{D}_T \neq \mathcal{D}_S$ can be classified as one of two cases: 1) $\mathcal{X}_S \neq \mathcal{X}_T$, meaning feature spaces are different, which Ruder (2019) refers to as *cross-lingual learning* where we can learn a mapping from source to target domain such that $P_S(f(X_S)) = P_T(X_T)$. 2) $\mathcal{X}_S = \mathcal{X}_T$ but $P_S(X_S) \neq P_S(X_T)$, meaning feature spaces are same but marginal probability distributions are not, which is referred to as *domain adaptation*.

[8]Notice that in Ruder (2019), $\mathcal{T}$ also includes a prior distribution $P(Y)$

We see here domain adaptation is usually seen as a case of transductive transfer learning. However, in other related works, domain adaptation is seen as a broader concept, and the definition given here is only for unsupervised domain adaptation.

Domain adaptation

Some domain adaptation papers assume target labeled data D_T is also available in very small amounts (Ben-David et al., 2007; Jiang, 2008). The approaches that use only target unlabeled data D_{TU} are categorized as *Unsupervised Domain Adaptation* approaches, while those that use a very small amount of target labeled data are called *Supervised Domain Adaptation* approaches (Daumé III, 2007). Furthermore, those that use both target unlabeled and a very small amount of target labeled data are called *Semi-supervised Domain Adaptation* (Daumé et al., 2010). The difference with semi-supervised learning is the additional usage of source domain labeled data D_S. Thus, in this sense, domain adaptation also has some overlap with inductive transfer learning. Here, we will adopt the broader concept of domain adaptation, and will refer to domain adaptation as the case $\mathcal{X}_S = \mathcal{X}_T$ but $P_S(X_S) \neq P_S(X_T)$ such as in transductive transfer learning, but we are not limiting the usage of target labeled data. Therefore, here domain adaptation is a scenario that will use source data either labeled D_S, unlabeled D_{SU} or both and also target data either labeled D_T, unlabeled D_{TU} or both. Furthermore, we may also need to address $P_S(Y_S|X_S) \neq P_T(Y_T|X_T)$ in domain adaptation (Jiang and Zhai, 2007b; Zhang et al., 2015).

In this thesis, given the assumption that an on-going crisis event's labeled data is hardly available, we will focus on the unsupervised domain adaptation. More formally, in unsupervised domain adaptation, we have $D_S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{m_S}, y_{m_S})\} \subseteq \mathcal{X} \times \mathcal{Y}$ and target unlabeled data $D_{TU} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{m_{TU}}\} \subseteq \mathcal{X}$, we may in addition have source unlabeled data $D_{SU} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{m_{SU}}\} \subseteq \mathcal{X}$, where $P_S(X_S) \neq P_S(X_T)$ and usually the size of the unlabeled dataset is large. The goal is to learn a good hypothesis function $h(\cdot)$ or $P_T(Y_T|X_T)$ for the target domain. For our classification tasks, we assume tweets of different disasters including different types of disasters still share some characteristics, so we assume that

$P_S(Y_S|X_S) = P_T(Y_T|X_T)$ in most of the cases. For example, intuitively donation related tweets or infrastructure damaged related tweets have very similar content in different disasters, thus we can assume that $P_S(Y_S|X_S)$ and $P_T(Y_T|X_T)$ are the same. Finally, while our discussion so far is based on only one source domain, we can also extend this to multi-source domain adaptation.

Multi-source domain adaptation

Currently, many research papers about domain adaption focus on single source domain adaptation. But in many cases, such as sentiment analysis and our disaster tweet classification problem, multiple sources may be available. To formalize the multi-source domain adaptation, we only need to change the number of sources from one to many in our notations. Formally, suppose we have S sources and $\mathcal{D}_{S_i}$ denotes the ith source domain. Let $S_i = \{(\mathbf{x}_1^{(i)}, y_1^{(i)}), \ldots, (\mathbf{x}_{s_i}^{(i)}, y_{s_i}^{(i)})\}$ be the ith source labeled sample. Our goal is to learn a good hypothesis $h(\cdot)$ on the target domain with help from these multi-source domain data and target unlabeled data.

Other related concepts

We should note that domain adaptation is different from *semi-supervised learning* in that the latter uses labeled and unlabeled data from the same domain. Furthermore, domain adaption is different from *multi-task learning* in that the latter learns several tasks simultaneously, while domain adaptation focuses on learning the target task using knowledge from same task of one or more different domains.

There are other concepts related to domain adaptation, and also other perspectives regarding domain adaptation distribution differences. For example, the case that assumes $P_S(Y_S|X_S) = P_T(Y_T|X_T)$ but $P_S(X_S) \neq P_T(X_T)$ is also termed *covariate shift* as a particular case of sample selection bias (Huang et al., 2006). The case when $P_S(X_S|Y_S) = P_T(X_T|Y_T)$ but $P_S(Y_S) \neq P_T(Y_T)$ is sometimes called *class imbalance* (Jiang, 2008), *prior probability shift* or *target shift* (Zhang et al., 2015).

Chapter 3

Literature review

In this chapter, we present the different approaches to domain adaptation through a comprehensive literature review in Section 3.1 followed by a literature review of social media crisis data classification in Section 3.2.

3.1 Domain adaptation approaches

In general, there are several different ways domain adaptation algorithms are used to align source and target domain distributions. *Inductive transfer learning* approaches have been organized by Pan and Yang (2010) into four different categories: *instance transfer, feature transfer, parameter transfer and relational knowledge transfer*, while only the first two categories are appropriate for *Transductive transfer learning*. As we discussed in Section 2.5, there are supervised and semi-supervised domain adaptation approaches, so these approaches somehow have overlap with *Inductive transfer learning*. Supervised and semi-supervised domain adaptation approaches can also be categorized into the four categories mentioned above. However, these categories are not exclusive from each other, so here we will only discuss research that is closely related to our work, especially unsupervised domain adaptation on text classification tasks. We put these works into four categories[1]: 1) Parameter based

[1]Note that these categories are listed for the convenience to follow our work. Just as definition of domain adaptation is not always consistent, the categories of the approaches can also vary. Other researchers may

approaches, which use some priors from the source domain to inform the target domain. 2) Instance based approaches, which weigh or select either only source labeled instances or both source labeled and target unlabeled instances with different methods to align the distributions of source and target domain. 3) Feature based approaches, where we change the feature representations of source and target to make them similar. 4) Pre-training language models and fine-tuning based, where we will use pre-trained language models and fine-tuning for the current tasks. We can also combine these different types of approaches to build a hybrid domain adaptation model.

3.1.1 Parameter and instance approaches

Most domain adaptation algorithms are designed around one question, which is how to bring the source labeled data close to the target data with help from target unlabeled data, so that the distribution can be better aligned. Instance based adaptation approaches try to decrease the domain shift between source and target domains by re-weighting source and/or target instances based on different criteria. For example, the distribution of the source domain can shift towards the distribution of the target domain by giving higher weights to source instances that are closer to the target unlabeled instances. Parameter based adaptation approaches are more related to multi-task learning, but they can be easily adapted to domain adaptation. Such approaches assume that related tasks share some parameters and prior distributions of hyper-parameters. However, from a probability perspective, approaches that estimate the target probability parameters with help from source data can also be considered a type of parameter based approach, which are sometimes achieved with a EM style process involving instance selection or weighting.

EM/Self-training and Co-training

Recall that an Expectation Maximization (EM) algorithm is used in statistical models to find maximum likelihood estimates of parameters iteratively (Dempster et al., 1977). Combined

put the approaches discussed here into different categories.

with machine learning classifiers that also rely on probability distribution estimations, such as Naive Bayes, EM can be used to improve the classifiers with unlabeled data. This approach has been used for many machine learning tasks, including text classifications (Nigam et al., 2000). Self-training (ST) (Yarowsky, 1995) labels the unlabeled data and converts the most confidently predicted document of each class into a labeled training example. This iterates until convergence similar to EM. Co-training algorithms (Blum and Mitchell, 1998) learn two or more functions based on distinct subsets of the features. To predict the labels, co-training trains these distinct functions both to fit the labeled examples and also to agree on their predictions for unlabeled examples. EM, self-training, co-training and also a hybrid of EM and co-training (co-EM) are being compared for text classification task in Nigam and Ghani (2000), where co-training performs well if the feature set independence assumption is valid. These approaches are being adapted to domain adaptation problems as well, and are sometimes referred to as self-labeling approaches. Below, we will review some papers related to these approaches and also some papers that are closely related to our proposed domain adaptation approach in Chapter 5, a weighted Naive Bayes classifier with self-training/EM.

Dai et al. (2007) proposed a domain adaptation algorithm, based on the Naive Bayes classifier and EM, to classify text documents from Newsgroups, SRAA, and Reuters into top categories. Experimental results showed that this algorithm performed better than supervised algorithms based on either Support Vector Machine (SVM) or Naive Bayes classifiers. Tan et al. (2009) proposed a weighted version of the multinomial Naive Bayes classifier combined with EM for sentiment analysis. Their algorithm filters out domain-specific features from the source domain, by keeping only the top-ranking features that have similar probabilities in both source and target domains. In the first step, a Naive Bayes classifier is trained on the source data and used to label the unlabeled data from the target domain. In subsequent iteration, the EM algorithm is used with a weighted combination of the source and target data to train a new Naive Bayes classifier. Specifically, in the maximization (M) step, the prior and likelihood are calculated, and in the expectation (E) step, the posterior is calculated for the instances in the target data. These steps are repeated until convergence, with the weight shifting from the source to the target domain, iteration by iteration. In

our experiments, we use Bernoulli Naive Bayes classifier instead, and we use target domain features to represent both source and target data.

Viswa Mani Kiran Peddinti (2011) used domain adaptation to perform sentiment classification of tweets. Given a source dataset, in addition to target labeled data, they proposed two methods to identify source instances that could improve the classifier for the target based on EM and Rocchio SVM. Namely, at each EM iteration, they first used target labeled data to classify source instances, then selected the most confident source instances to add back to the training set. Therefore, this method requires a small amount of target labeled data.

Herndon and Caragea (2015) proposed an approach like the one in Tan et al. (2009) for the task of splice site prediction. They used a weighted Naive Bayes classifier, and three methods for incorporating the target unlabeled data: EM with soft-labels, ST with hard-labels, and also a combination of EM/ST with hard-labels for the most confidently labeled instances in the current target unlabeled data, and soft-labels for the other instances. They found that for the task of splice site prediction, EM with soft-labels gives better classifiers than the other two methods, contrary to what has been observed in text classification (Nigam and Ghani, 2000), where ST with hard-labels gives better results. Here, a small amount of target labeled data is used which is different from our approach that no target labeled data is used.

In Li et al. (2015), we proposed a domain adaptation approach based on the iterative EM algorithm and a weighted Naive Bayes classifier, for identifying disaster relevant tweets. In this approach, a classifier is learned at each iteration, and used to label the target unlabeled data. Subsequently, the target data, with probabilistic soft-labels assigned by the current classifier (e.g., $p(+|d) = 0.7$ and $p(-|d) = 0.3$ for an instance d), are combined with the labeled source data and used to train the classifier at the next iteration. The original classifier is trained from source data only. The process continues for a fixed number of iterations, or until convergence, by slowly giving more weight to soft-labeled target data during training.

Some other variations of self-training have also been proposed (Guo et al., 2012). Similar to the EM strategy, our proposed self-training based domain adaptation approach (Li et al., 2018a) is an iterative approach that uses a weighted Naive Bayes classifier to combine source

and target data. Just like the EM approach, it starts by learning a supervised classifier from source data only, and uses that classifier to label the target unlabeled data. However, instead of adding all the target data with probabilistic soft-labels to the training set for the next iteration as in EM, in self-training only the most confidently classified data are added to the training set, with hard (e.g., +/- or 1/0) labels.

Chen et al. (2011a) adapted co-training for domain adaptation (CODA) using pseudo-multiview regularization that they proposed in (Chen et al., 2011b) to split the features into two mutually exclusive views. CODA slowly adapts the training set from the source distribution to the target, both data-wise and feature-wise. To shift the distribution of the training data from source to target, CODA gradually adds inputs from the target domain to the training set using co-training. Meanwhile, it includes a feature selection component to shift the features used by the model from source-heavy to target-heavy features. Combining these two strategies, CODA significantly outperforms self-training as well as some feature based domain adaptation approaches we will discuss later on a sentiment analysis task.

Some recent papers also use self-training in a similar setting to domain adaptation and achieve satisfactory results. For instance, self-training has been used for zero-shot text classification, where for some classes there are no labeled examples but other classes may have some labeled examples. Ye et al. (2020) proposes a self-training based method where they use a reinforcement learning framework to select unlabeled instances for zero-shot classification. Xie et al. (2020) proposes self-training with noisy student to improve image (ImageNet) classification in a semi-supervised setting.

Instance weighting and selection

Instance weighting methods aim to approximate target distribution $P_T(\mathbf{x}, y)$[2] through estimating $\frac{P_T(\mathbf{x},y)}{P_S(\mathbf{x},y)}$ and $P_S(\mathbf{x}, y)$ because $P_T(\mathbf{x}, y) = \frac{P_T(\mathbf{x},y)}{P_S(\mathbf{x},y)} P_S(\mathbf{x},y)$. We can estimate $P_S(\mathbf{x}, y)$ with the source labeled data, but $\frac{P_T(\mathbf{x},y)}{P_S(\mathbf{x},y)}$ is unsolvable. However, we can approximate $\frac{P_T(\mathbf{x},y)}{P_S(\mathbf{x},y)}$ by adding each source example with a weight which we can get with small amount of target labeled and target unlabeled data. Based on Bayes' rule, we can further transfer $\frac{P_T(\mathbf{x},y)}{P_S(\mathbf{x},y)}$ into two

[2]here means $P_T(X_T = \mathbf{x}, Y_T = y)$, the joint distribution of target domain, similar for P_S.

parts: 1) $\frac{P_T(y|\mathbf{x})}{P_S(y|\mathbf{x})}$ and 2) $\frac{P_T(\mathbf{x})}{P_S(\mathbf{x})}$[3]. If we assume the first term is 1, meaning $P_T(y \mid \mathbf{x}) = P_S(y \mid \mathbf{x})$, then we just need to focus on the ratio of the marginal distribution, or the second term. If we can estimate this ratio for each instance from source and target, then we can solve the adaptation problem. There are many different methods described in the literature to estimate this ratio. Some works set these as parameters that are tuned and learned during the training step, some works estimate the weight for each example through minimizing the distance such as KL divergence between the weighted source domain examples and target domain examples. Our self-training with Naive Bayes approach also involves a weighting shift from source examples to target unlabeled examples, and therefore our self-training with Native Bayes approach is related to instance weighting approaches.

Jiang and Zhai (2007b) introduced an instance weighting framework for domain adaptation for use in NLP tasks (POS tagging, name entity and spam filter). Besides source labeled data D_S. They assume both a small set of target labeled data D_T and also a large amount of unlabeled target data D_{TU} are available. From the probability perspective, they proposed three ways to approximate the target input spaces $\mathcal{X}$, using D_S, D_T, D_{TU} respectively. They assigned three parameters to control the contributions from these three approximations. They also assigned a parameter α_i to indicate how likely $P_T(y_{T_i}|\mathbf{x}_{T_i})$ is to be close to $P_S(y_{T_i}|\mathbf{x}_{T_i})$. Large values for α_i indicated that we could trust this instance and therefore this parameter could help remove misleading training instances. Another parameter β_i was also assigned to source instances to approximate $\frac{P_S(X_S)}{P_T(X_T)}$, and finally a parameter γ_k similar to the parameter in bootstrapping semi-supervised learning was set to be weight for a target unlabeled instance in D_{TU}. Those confidently predicted instances were set to 1, others to 0. The experiments on NLP tasks (POS tagging, name entity and spam filter) were comparable to other methods.

Huang et al. (2006) estimated $\frac{P_S(\mathbf{x})}{P_T(\mathbf{x})}$ by matching the means between the source domain and target domain data, which is referred to as Maximum Mean Discrepancy (MMD) in other works. The main idea is to minimize the distance between two distributions based on a

[3]Some works may also focus on 1) $\frac{P_T(\mathbf{x}|y)}{P_S(\mathbf{x}|y)}$ and 2) $\frac{P_T(y)}{P_S(y)}$, assuming the first term is 1, then focusing on the class imbalance problem.

mapping to a Reproducing Kernel Hilbert Space (RKHS). Sugiyama et al. (2007) and Tsuboi et al. (2009) used KL divergence as distance function to minimize and to determine the instance weights. Søgaard and Haulrich (2011) and Plank et al. (2014) used the probabilities produced by a domain classifier that was trained separately as instance weights.

Unlike weighting each instance in instance weighting approaches, instance selection approaches aim to just pick the instances that will help the algorithm to learn a good target domain function. Instance weighting and instance selection are related similarly to the relationship between EM and self-training methods we discussed above, where with EM we assign a soft-label and with self-training we assign a hard label. However, here the instance selection focuses more on source instances rather than target unlabeled data, especially in multi-source domain adaptation problems. Most of the instance selection approaches will first estimate the weights or scores for the instances, and then select the instances based on some threshold. For example, Ruder et al. (2017b) studied different data selection metrics for sentiment analysis under a multi-source domain setting, on the domain level (the close domains), on the instance level (the close instances), and on the instance subset level or feature subset (the close instances based on partial features). Ruder et al. (2017a) proposed a data selection metric called maximum cluster difference (MCD), which is the absolute difference in similarity of an instance's representation with the cluster centroids of the positive and negative classes. This measure is inspired by the clustering assumption in semi-supervised learning and motivated by the observation that incorrect predictions are frequent along the decision boundary. This measure was also used in Guo et al. (2018) for multi-source domain adaptation where they proposed a mixture-of-experts approach to explicitly capture the relationship between a target example and different source domains.

In (Mazloom et al., 2018) and the extended version (Mazloom et al., 2019), we also proposed to select source instances based on a matrix factorization and k-nearest neighbors algorithm. The proposed hybrid adaptation approach is used to select a subset of the source disaster data that is representative of the target disaster. The selected subset is subsequently used to learn accurate supervised or domain adaptation Naive Bayes classifiers for the target disaster.

3.1.2 Feature representation approaches

Feature representation based approaches aim to bring the source domain and target domain closer through change feature representations. For example, this can be done by projecting data into a low-dimensional space with singular value decomposition (SVD) or latent Dirichlet allocation (LDA), or using other representation approaches. Many domain adaptation approaches with deep learning models often involve representation learning. We will first review some traditional methods, which are not based on deep learning models. Then, we will review the concepts of an autoencoder and a domain adversarial network, as well as discuss some papers based on the idea of domain adversarial networks. Since we study pre-trained word embeddings and sentence encoding models for crisis tweets representations, we will discuss some of those works here. Just as with transfer learning, some other researchers may put domain adaptation approaches with word embedding into other categories, such as sequential transfer learning, as we discussed previously.

Traditional feature based approaches

Blitzer et al. (2006) introduced structural correspondence learning (SCL) to automatically induce correspondences among features from different domains for part-of-speech tagging domain adaptation task. Blitzer et al. (2007) extended SCL such that the selection of pivot features not only based on common frequencies, but also based on mutual information for use in sentiment analysis. Given D_S and D_{TU}, SCL first chooses pivot features based on their frequencies in both domains. It then uses linear pivot predictors to predict occurrences of each pivot in the unlabeled data from both domains to model the correlation between the pivot features and all other features. Each pivot is finally characterized by its weight vector. All such weight vectors then form the matrix which can be used to select the top principal predictors. Then, if such pivot features are chosen appropriately, the principal predictors can discriminate both domains well. However, in sentiment analysis, frequently occurring words may vary significantly for different domains. Therefore, they proposed to use mutual information to choose the pivot features that had high mutual information to

the source label. They further proposed to use a small amount of target labeled data D_{TL} to correct the misalignment of the pivot features. Experiments on four kinds of Amazon product reviews showed that with correcting misalignment, the model could always improve from the baseline. They also showed that $\mathcal{A}-$distance which was analyzed by Ben-David et al. (2007) correlated well with adaption loss with respect to the golden in-domain accuracy. This Amazon multi-domain dataset they published along with this paper has also become a standard popular dataset for domain adaptation on sentiment analysis.

In (Daumé III, 2007), the author assumes there is some target labeled data D_{TL} available in addition to source labeled data D_S. He proposed a very simple method for representing the source and target with duplicate features consisting of three versions: the general version, the source specific version and the target specific version. An instance from domain $\mathcal{D}_i$ will be represented by both its original features and the features specific to $\mathcal{D}_i$. All others features from other domains will be zero. As the author argued, in the single source scenario, if we were to apply this method to a kernelized version and then took the kernel as a measure of similarity, then data points (instances) from the same domain would be twice as similar as those from different domains. This would give the target domain twice as much as predictive influence as source instances. We can easily extend this method to multi-source domain adaptation. The experiments with the NLP sequence labeling tasks (named-entity recognition, shallow parsing or part-of-speech tagging) showed that this method outperform the baseline in most cases. Some of these experiments used multi-source on one target. In (Jiang, 2008), similarities were drawn between this method and some multi-task learning methods. Jiang and Zhai (2007a) also proposed a two-stage domain adaptation method where in the first stage they looked for a set of features generalizable across domains, and in the second adaptation stage they picked up useful features specific to the target domain.

Pan et al. (2010) proposed a spectral feature alignment (SFA) algorithm with an idea similar to SCL but aiming to align domain-specific words from different domains into unified clusters with the help of domain-independent words acting as a bridge. Then, the clusters can be used to reduce the gap between domain-specific words of the two domains, which can be used to train sentiment classifiers in the target domain accurately. SFA first selects domain-

independent words that are frequent in both domains and also not high mutual information on the source domain. It then constructs a bipartite graph to model the relationship between domain-specific and domain-independent features, and finally it adapts a spectral clustering algorithm on the feature bipartite graph to align domain-specific features. Compared to previous approaches, SFA can discover a robust representation for cross-domain data by fully exploiting the relationship between the domain-specific and domain-independent words via simultaneously co-clustering them in a common latent space.

Sun et al. (2015) also proposed a simple CORrelation ALignment (CORAL) to represent the source domain to better align with the target domain for use on object recognition and sentiment analysis. They assume only source labeled data D_S and unlabeled target data D_{TU} is available. CORAL aligns the input source distribution by re-coloring whitened source features with the covariance of the target distribution. They also use the Amazon product reviews dataset. Despite the simplicity of this method, it can produce results comparable to other, more complex algorithms (TCA, GFS and GFK) which we will not review here. The authors later extended CORAL to deep neural networks such as CNN (Sun and Saenko, 2016). They proposed to add CORAL loss to the hidden layers of the model in order to minimize the difference in second-order statistics between source and target feature activation. The CORAL loss is jointly trained with classification loss of the source labeled data. We adopted CORAL as a feature based approach and combined it with self-training on the crisis tweets classifications.

Pre-trained word embeddings

With standard bag-of-word fixed vocabulary representations, the similarities between words are not considered. Although simple, such representations have their limitations when it comes to domain adaptation problems. When the source and target domains do not share too many features, such bag-of-words representations may limit the transfer from source to target domains. Distributed words representations, i.e., word embeddings, which capture similarities by representing words in a semantic vector space, can be helpful to address this

problem. With each word as a vector, similarities between two words can be calculated through cosine-similarity or other measurements. Then, the similarities between tweets from source and target domains, which use different words but similar semantic meaning to humans, can be captured through word similarities. Therefore, in this research, we also study different ways to utilize word vector representations.

Just like the deep learning concepts, the idea of distributed word representations is not new, but has become more popular given the more powerful computing resources currently available. The most popular and successful works on word embeddings are Word2Vec by Mikolov et al. (2013a,b) and GloVe by Pennington et al. (2014). The models from both papers are trained on a large corpus, and the trained models of word vectors are publicly available. GloVe is similar to the Word2Vec model but uses global word counts and co-occurrences of word pairs.

Given a tweet, there are many ways it can be represented using word embeddings. The simplest way is to tokenize the tweets and represent the whole tweet as a vector by averaging the word embeddings per each dimension. We then end up with a vector having the same dimensions as the word vectors. Alternatively, in addition to averaging the word vector embeddings, we can also keep the minimum and maximum value of each dimension and concatenate them. In this case we end up with a vector having three times the dimensions of the word vectors. We can also use weighted averaging (Arora et al., 2017), which, although very simple, has been shown to be hard to beat for sentence embedding in sentence similarity tasks.

Similar to word embedding, there are also sentence encoding models that learn to build better representations on a sentence level, such as (Cer et al., 2018; Conneau et al., 2017). The models are trained on a natural language inference dataset, and are trained to capture the semantics when producing the sentence embedding. We use these models to generate sentence embedding and then build a supervised classifier or domain adaptation classifier on top of that. We can also represent a tweet as a matrix using deep learning models like Convolutional Neural Network (CNN) or Long short-term memory (LSTM).

Existing NLP works that compare different word embeddings aim to evaluate the qual-

ity of the embeddings across different NLP tasks (Baroni et al., 2014; Nayak et al., 2016; Schnabel et al., 2015). Evaluation approaches can be grouped into two categories: intrinsic evaluation and extrinsic evaluation (Schnabel et al., 2015). Intrinsic evaluations measure the quality of word embeddings by directly computing the correlation between semantically and geometrically related terms, usually through pre-collected inventories of query terms (Schnabel et al., 2015). For instance, Word2Vec is evaluated on a word similarity task in the original paper in which it was introduced. Baroni et al. (2014) and Faruqui and Dyer (2014) also focus on intrinsic evaluations using a variety of query inventories.

In extrinsic evaluations, word embeddings are used as input features to a downstream task, and the evaluation of the word embeddings is done according to the performance metrics specific to that task (Schnabel et al., 2015). For instance, GloVe embeddings are evaluated on part-of-speech tagging and named-entity recognition tasks (Pennington et al., 2014). Nayak et al. (2016) also proposed to evaluate word embeddings using a standardized suite of characteristic downstream tasks, so that the evaluation is more likely to be generalized to real-world applications of the embeddings. Together, a thorough intrinsic evaluation and a limited extrinsic evaluation of word vectors (Schnabel et al., 2015) show that the performance on two downstream tasks (noun phrase chunking and sentiment classification) is not consistent, and may not be consistent with intrinsic evaluations either. The authors suggest that training specific embeddings to optimize a specific objective is generally better for downstream tasks. This is one reason we compare pre-trained embeddings with crisis-specific embeddings on crisis tweet classification. Schnabel et al. (2015) showed, in the context of sentiment classification of movie reviews, that CBOW (a Word2Vec model) was better than GloVe and some other distributed word representations. In their study, the word embeddings for sentiment classification are used to generate embedding-only features for each movie review by computing a linear combination of word embeddings weighted by the number of times that word appeared in the review.

Other prior studies that used word embeddings on tweet classification tasks also generated the vector representations of tweets by averaging the word embedding vectors along each dimension for all the words in a tweet (Boom et al., 2016; Wang et al., 2015; Yang

et al., 2016). The average representation was usually compared with the weighted average word embedding representation, and/or with approaches that use the word embeddings in deep learning models such as Convolutional Neural Networks (CNN). For example, Boom et al. (2016) focused on learning short text representations by averaging each dimension of the word embeddings. They also experimented with averaged embeddings concatenated with minimum and/or maximum aggregated embeddings, and propose a weight-based approach that performs better on their semantically related and non-related pairs of words from Twitter data. Yang et al. (2016) studied the effect of the configuration used to train and generate word embeddings on a Twitter election classification task, where average word embedding representations of tweets, used with the SVM classifier, were compared with CNN models. The results suggested that the CNN models outperformed the SVM models.

While many previous studies have focused on text representations through the means of word embeddings or sentence encoders, the research on the usability of word/sentence embeddings for crisis tweet representation and classification is limited. Nguyen et al. (2016) used Word2Vec embeddings to initialize CNN models trained to classify crisis tweets. They also used crisis-specific Word2Vec embeddings trained on a corpus with approximately 60,000 tweets, and showed that the crisis-specific embeddings were slightly better than the pre-trained embeddings. We also compared pre-trained and crisis-specific embeddings, and experimented with comparing a larger variety of word embeddings used subsequently with supervised classifiers. Furthermore, our training corpora (about 5.8 million tweets) for crisis-specific embeddings is much larger than the one used by Nguyen et al. (2016).

Autoencoders

An autoencoder tries to encode the input through an encoder function and subsequently reconstructs the input through a decoder. It is trained to minimize the reconstruction error, so it is essentially an unsupervised method. If the features of the input are correlated, then the hidden layer maybe able to discover the correlation and often gives a reduced representation of the original input. Thus, domain adaptation from source to target can be

achieved by changing the representations through the encoder trained on source data and target unlabeled data. One alternative to standard autoencoder is Denoising Auto-encoder (Vincent et al., 2008) or DAE, in which the input vector is stochastically corrupted and the model is trained to minimize a denoising reconstruction error. The benefit of DAE comes from learning a more robust feature representation. An autoencoder or DAE can be stacked, with the second encoder taking the output of the first encoder as input etc., to form a stacked autoencoder.

Glorot et al. (2011) first proposed Stacked Denoising Auto-encoders (SDA) for domain adaptation problems on sentiment analysis. They used SDA to get the latent representations of both the source and target domains, then with those representations a supervised SVM was trained for classification. To address the limitations of the high computational cost and lack of scalability to high-dimensional features of SDA, Chen et al. (2012) proposed marginalized Stacked Denoising Auto-encoders (mSDA) which do not require stochastic gradient descent or other optimization algorithms to learn parameters but still achieve almost identical performance. There are also other autoencoders, such as those proposed in (Zhou et al., 2016; Zhuang et al., 2015). Instead of putting the source and target domain instances together to the autoencoder, Zhou et al. (2016) proposed a Bi-Transferring Deep Neural Networks (BTDNNs), which essentially trains an autoencoder that has one encoder but two decoders to transfer the source domain examples to the target domain, and also to transfer the target domain examples to the source domain.

Ziser and Reichart (2017) proposed a model that combines SCL and autoencoders (AE-SCL), which is a three-layer neural network that learns to encode the non-pivot features of an input example into a low-dimensional representation, so that the existence of pivot features in the example can be decoded from that representation. The low-dimensional representation is then employed in a learning algorithm for the task. They later also proposed the Pivot Based Language Model (PBLM) in (Ziser and Reichart, 2018). This model is: 1) aware of the structure of its input text and 2) outputs a representation vector for every input word, so that the classification later can be performed with deep learning models like CNN or LSTM. PBLM is a sequential neural network (LSTM) that operates very similarly to

LSTM language models (LSTM-LMs). The main difference is that while for every input word LSTM-LMs outputs a hidden vector and a prediction of the next word, the output of PBLM is a hidden vector and a prediction of the next word if that word is a pivot feature, otherwise the output is given a generic NONE tag.

Domain adversarial neural networks

Another line of research on domain adaptation follows the popularity of Generative Adversarial Nets (GANs) (Goodfellow et al., 2014), which are called domain adversarial neural networks (DANN) (Ajakan et al., 2014; Ganin and Lempitsky, 2015; Ganin et al., 2016). Different from autoencoder based methods, which learn the feature representations and then any supervised classifier can be applied on top of those feature representations, domain adversarial learning combines classification and feature learning within one training process and therefore can be used with any existing feed-forward architecture that is trainable by backpropagation. The framework focuses on learning features that are discriminative but at the same time domain-invariant. This is achieved with two classifiers on these features: 1) a label classifier that predicts labels; 2) a domain classifier that discriminates between the source and target domains during training. The parameters are optimized to minimize the loss of the label classifier while maximize the loss of the domain classifier, so that the feature representations learned for source and target domain are hard to distinguish. A new gradient reversal layer is added to a domain classifier so that these two classifiers can be trained together with standard backpropagation.

The neural network in DANN is simply a one layer neural network, so there are many later studies that use other more complex neural network architectures with the domain adversarial idea. For example, Li et al. (2017b) proposes an adversarial cross domain model for sentiment classification with memory network. Additionally, Du et al. (2020) uses a pre-trained language model BERT that we will later discuss. It should be noted that the domain adversarial training idea has already been extend to multi-source domain adaptation (Zhao et al., 2017).

3.1.3 Pre-trained language models and fine-tuning based approaches

The default task for language models is to predict the next word given the past sequence. This idea, framing a supervised learning task in a special form to predict only a subset of information using the rest subset, is sometimes referred to as *self-supervised learning*. We can change the default task to be predicting any "missing" part of the input or we can bring in more tasks like predicting the next sentence.

The idea of fine-tuning pre-trained models can be considered to have originated with computer vision. In computer vision models, for example a CNN for image classification, the intuition of the filters is that it may be able to detect artifacts such as an edge at the first layer and then some simple shape in upper layers. After this model is trained, the filters have learned to detect different levels of an object in the image. Therefore, many models in computer vision are not trained from scratch but from a pre-trained model such as a model trained on ImageNet and then fine tuned to the current task. This kind of approach has also been used in domain adaptation or transfer learning in computer vision. With computational power readily available, language models trained with very large corpora and the deeper models introduced recently, there have been a significant impact on transfer learning in NLP. Similar to pre-trained image classification models, language models can capture different levels of feature representations. For instance, a word embedding layer captures word similarity, then subsequent lower layers may capture local syntax, while upper layers may capture semantics. Similar to pre-trained word embedding for domain adaptation, intermediate representations in pre-trained language models will ideally capture the similar semantic and/or structural meanings between source domain and target domain, allowing us to then fine tune the model to get a target domain classifier.

Dai and Le (2015) were the first to propose the use of unlabeled data to train a language model as a pre-training step to help the supervised text classification task. Their work is in a semi-supervised setting. Peters et al. (2018) proposed to use a language model objective to get deep contextualized word embedding which they named the ELMo (Embeddings from Language Models) representation. They are also the first to show that a pre-trained language

model is useful for several different NLP tasks. They use a character level CNN followed by a language model based on Bi-LSTM. Howard and Ruder (2018) proposed Universal Language Model Fine-tuning (ULMFiT), which pre-trained a language model on a large general-domain corpus and then fine-tuned it on the target task to retain previous knowledge. To avoid catastrophic forgetting during fine-tuning, they proposed discriminative fine-tuning, slanted triangular learning rates and gradual unfreezing. Radford et al. (2018) proposed a deeper model with larger training data and demonstrated large gains on several NLP tasks with generative pre-training of a language model and fine-tuning. This model is usually referred to as GPT for generative pre-training. They later also scaled the model up to GPT-2 as a deeper model (1.5 billion parameters, 10 times of GPT) with more than 10 times the amount of data. The even more powerful model GPT-3 is trained to improve few-shot performance without a fine-tuning process. Sometimes, the results are competitive with fine-tuning approaches.

Devlin et al. (2018) proposed a new language model called Bidirectional Encoder Representations from Transformers (BERT). Unlike the previous language models whose main tasks are to predict the next word, which just go in one direction, BERT alleviates this unidirectionality constraint by introducing a masked language model (MLM) pre-training objective. The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context. The MLM objective enables the representation to use the left and the right context, so as to pre-train a deep bidirectional model. In addition, they also jointly train another task, next sentence prediction. Both GPT and BERT are based on a network architecture, Transformer, that is based on solely on the attention mechanisms (Vaswani et al., 2017). There are other variants also based on the previous language models, including RoBERTa, XLNet and ALBERT. Those models try to improve the previous language models either with a more efficient training process or reducing the parameters of the model.

Since its initial proposal, BERT has been used in many different tasks and settings including domain adaptation. Nguyen et al. (2020) introduced BERTweet, a large-scale pre-trained language model for English Tweets that was trained with the same architecture as

BERT but with the training procedure from RoBERTa. In domain adaptation of sentiment analysis, Du et al. (2020) investigated how to efficiently apply the pre-training language model BERT on the single source unsupervised domain adaptation for sentiment analysis of the Amazon multi-domain data. They first take a pre-trained BERT model and further train it with two tasks: 1) domain-distinguish task, where instead of predicting the next sentence in original BERT model, they predict whether two sentences are from the same domain or from different domains; 2) target domain MLM, where they further train the model with MLM on a target domain to inject target domain knowledge. With the post-trained BERT model, they further proposed to use adversarial training. And this combination of post-training and adversarial training is shown to be better with just fine-tuning or adversarial training with vanilla (default pre-trained) BERT model.

3.1.4 Multi-source domain adaptation approaches

Similar to the case of single source domain adaptation, multi-source domain adaptation algorithms can also be categorized into different groups, e.g. feature representation or instance based approaches. We have introduced some relevant papers in previous sections. The difference is that we have several sources, each with different distribution from the target domain. How to combine these sources to get a better model represents the main challenge. We can simply combine all sources as one, we can weight the sources, or weight the hypothesis built from each source and then form a target hypothesis. Multi-source may also apply instance selection or weighting like in single source domain adaptation, some of which we have briefly mentioned in the previous discussion, for example (Guo et al., 2018; Ruder et al., 2017a,b). We will briefly review some other works here.

Mansour et al. (2008) provided a theoretical analysis of the linear combinations of hypotheses (or classifiers) from the sources, and proved that the standard convex combinations of the source hypotheses may perform poorly. Instead, combinations by the sources distributions benefited from favorable theoretical guarantees. Blitzer et al. (2008) provided learning bounds for algorithms that minimized a convex combination of source and target empirical

risk through theory. These bounds explicitly model the inherent trade-off between training on a large but inaccurate source data set and a small but accurate target training set. The theory in this paper is cited by many other papers and be extended to multi-source domain adaptation setting in (Zhao et al., 2017). Based on the theory, Zhao et al. (2017) proposed Multi-source Domain Adversarial Network (MDAN) that has same general architecture as domain adversarial neural network but consider each source and target pair during training.

Duan et al. (2009) developed a multi-source domain adaptation method called Domain Adaptation Machine (DAM) to learn a target classifier by leveraging a set of pre-computed classifiers. The idea is that based on the smooth assumption, the target classifier should have similar decision values with the auxiliary classifier of the source domains on the unlabeled target data. They use a parameter γ_i as the weight for measuring the relevance between the ith source domain and the target domain, which is set based on Maximum Mean Discrepancy (MMD) as distances of (D_S, D_T). Then they define a data-dependent regularizer, and minimize it together with the structural risk functional of Regularized Least-Squares (RLS), Least-Squares SVM (LS-SVM). In other words, they combine source SVM classifiers by weighting each source using a distance measured by MMD.

Zhang et al. (2015) studied multi-source domain adaptation from a casual view. They use causal models to represent the relationship between X and Y. For example, in clinic diagnosis, disease Y usually will cause a symptom X. Therefore, casual background knowledge of X and Y can help us choose the corresponding algorithms. Even without such knowledge, available multi-source domain distributions may allow us to find a causal structure of the target. So, from the causal point of view, they consider four possible situations: 1) if $X \to Y$, $P(X)$ changes with the domain but $P(Y|X)$ stays the same; 2) if $X \to Y$, $P(X)$ may stay the same or change and $P(Y|X)$ changes with the domain; 3) if $Y \to X$, $P(Y)$ changes with domain but $P(X|Y)$ stays the same; 4) if $Y \to X$, $P(Y)$ may stay the same or change, but $P(X|Y)$ changes with domain. They further focused on the last situation where both $P(Y)$ and $P(X|Y)$ changes with domain. If we assume $P_T(X_T|Y_T)$ is linear mixture of $P_{S_i}(X_S|Y_S)$, then we could reconstruct $P_T^{new}(X)$ from $P_{S_i}(X_S|Y_S)$ with some parameters to approximate $P_T(X_T)$ according to Bayes rule. Thus, we can estimate the parameters by minimizing the

maximum mean discrepancy (MMD) between $P_T^{new}(X)$ and $P_T(X_T)$. Then, given the parameters estimated above, we can use different weighting methods to construct the target domain classifier. We can accomplish this by reweighting the source instances or by general model, or by a weighted combination of source classifiers. They also considered the case of a weighted distribution of hypotheses in (Mansour et al., 2008). The experiments on the Amazon product reviews showed that weighting samples and combination of classifiers give better accuracy when compared with other methods such as pool-SVM.

Due to the ability of feature representation methods to transfer to novel adaption problems (Hoffman et al., 2012), some feature representation methods used in single domain adaptation cases can be easily extended to multi-source domain adaptation. Such a case is presented by (Daumé III, 2007). In object detection, Hoffman et al. (2012) also extended their previous work on single transform domain adaptation which maps both source and target data into kernel space to multi transform domain adaptation. In this method, they first consider a similarity function between source and target parameters by a matrix W i.e. $sim_W(X_s, X_t) = \phi_s(X_s)W\phi_t(X_t)$ where ϕ is the matrix of data mapped to kernel space for source and target, respectively. Then W can be learned with a regularized optimization problem with a loss function depending on the similarity function, and finally a classifier such as SVM can be used to predict the test data. For the multi-source case, they first use the single feature transformation W_i for each source. The label of a new test instance then amounts to a weighted sum of the probability of a class, given the test instance is from a particular domain where the weights are the probabilities that the test instance belongs to each domain. So, for this method to work, all of the source domains should be known. However, there are cases where we will have training data whose domains are unknown. Therefore, they also propose a novel constrained clustering method that can discover latent domains.

3.2 Social media crisis data research

Work that has studied the value of social media in emergencies, known under the name of *Information and Communications Technologies (ICT)* in the early days, can be dated back to the 9/11 terrorist attacks in 2001 (Reuter and Kaufhold, 2018; Reuter et al., 2018). Since then, many practices have proved the value of social media and of the data extracted from it during emergencies. Research surrounding social media crisis analysis spans over multiple disciplines, from management science such as emergency management to information systems and computer science field. There is a significant amount of research on this topic with different focuses and viewpoints, including a management or information systems point of view. We will only review the research that is focused on machine learning methods with social media crisis data, specifically tweets in this thesis.

To optimize current organizational mechanisms in terms of speed, efficiency and knowledge, supervised machine learning algorithms have been used to help crisis responders sift through the large amounts of crisis data, and prioritize information that may be useful for response and relief (Ashktorab et al., 2014; Beigi et al., 2016; Caragea et al., 2014; Gao et al., 2011; Imran et al., 2015; Kumar et al., 2014; Terpstra et al., 2012; Yin et al., 2012). In the context of disaster response and rescue, there are several studies that have applied machine learning and natural language processing (NLP) methods for disaster management (Kumar et al., 2014; Purohit et al., 2014; Sakaki et al., 2010; Terpstra et al., 2012). In particular, several works have studied supervised learning algorithms in regard to transferring information from a prior source disaster to a current target disaster (Imran et al., 2013a, 2016; Verma et al., 2011). Other than works from our group, there are also works on domain adaptation approaches on crisis tweets from several other research groups. We will review them here.

Verma et al. (2011) used natural language processing techniques together with machine learning algorithms, Naive Bayes and Maximum Entropy, to identify situational awareness tweets during crisis events. They use data from four crisis events: the Red River Floods in 2009 and in 2010, the Haiti Earthquake in 2010 and the Oklahoma Grass Fire in 2009. They first build two supervised classifiers with Naive Bayes and Maximum Entropy to classify

situational awareness tweets from each of the four crisis events, respectively. Subsequently, they also study how well the classifiers performed across the four events. They find that the classifiers generalized well across the Red River Flood 2009 and Red River Flood 2010 events, but not for the other two events. For example, the performance (measured as accuracy) is poor when using the classifier learned from the Haiti Earthquake data to classify the Oklahoma Grass Fire data and vice versa, because these two types of events differ from each other in many aspects.

Imran et al. (2013a) performed similar experiments with two disasters, namely the Joplin Tornado (as source) and Hurricane Sandy (as target), to identify information nuggets using conditional random fields (CRF). After classifying different types of informative (casualties, donations, etc.) tweets with Naive Bayes classifiers, they use a sequence labeling algorithm, conditional random fields (CRF), to extract useful information, such as the number of casualties or the name of the infrastructure. They learn supervised classifiers either from source, or from source and 10% of labeled target data. They test these classifiers on all target data and the remaining 90% of target data, respectively, and compare the domain adaptation results with the results of the supervised classifiers learned from 66% of labeled target data, which is then tested on 33% of the target data. Their experiments show that only using source data results in a significant drop in the detection rate, while not significantly affecting the recall.

Imran et al. (2016) studied the usefulness of previous disaster tweets, and also the usefulness of incorporating data in different languages. They experiment with several pairs of disasters, earthquakes and floods, from different countries. They learn a Random Forest classifier from a source disaster to classify a target disaster. Their results also show that data from prior disasters of the same type as the current disaster can be very useful even across different languages.

While these works represent great steps towards using domain adaptation for disaster and crisis situations, the performances of the supervised classifiers used across different types of disasters or events are still poor, especially for domain specific tasks (e.g., identifying tweets relevant to a certain disaster). Therefore, deep learning models, and also domain adaptation

techniques that have been successfully used in text classification, have been proposed or adopted to disaster and crisis tweet classifications.

Caragea et al. (2016) explored the use of Convolutional Neural Networks (CNN) to classify informative tweets from six flood events. Their idea to identify overall well-fitting parameters for their models is similar to the idea presented in our paper. To implement this idea, they use three flood disaster datasets with labeled instances, tune parameters on one dataset and then test them on the other two test datasets to see how well the tuned parameters generalize.

Nguyen et al. (2016) used Convolutional Neural Networks (CNN) to classify crisis related tweets. They assume that some target labeled data is available and use two simple supervised domain adaptation techniques to combine prior source disaster data with current disaster labeled data during training. One technique is weighting the prior source disaster data, while regularizing the modified model. The other technique is simply selecting a subset of the prior source disaster tweets, specifically those that are correctly labeled by a target-based classifier. It is experimentally shown that CNNs with the simple instance selection domain adaptation technique gave better results. One drawback of these approaches is the requirement that some target labeled data is available.

Zhang and Vucetic (2016) proposed an approach that can be seen as a supervised domain adaptation which requires target labeled data. They also use the same dataset that is studied in this thesis. Their proposed approach first clusters words from unlabeled tweets, and then trains a logistic regression classifier on labeled disaster tweets represented with the word clusters as features. They vary the number of current disaster labeled instances and find that the performance is generally better with more labeled data. This result is similar to our result which suggests that the more source labeled data, the better the performance. However, we discover that for our domain adaptation approach, more source labeled data can help only up to the point where the performance stabilizes. They also vary the number of unlabeled instances from the current disaster and/or other source disasters, and even pre-trained a vocabulary based on word clustering. They find that more unlabeled data gives better results in general, with some exceptions. In our experiments, the performance

stabilized around 3000 target unlabeled instances for the disaster relevant vs. irrelevant task.

Alam et al. (2018a) proposed a domain adaptation approach that combines domain adversarial training and graph embeddings with a classification network. The adversarial training is used to reduce the distribution shift, while the graph embeddings are used to induce structural similarity between source and target data. Yao and Wang (2020) proposed to apply a domain-adversarial neural network on a sentiment analysis of tweets posted during hurricanes. Their method first retrieves hurricane-relevant tweets with a supervised trained Random Forest classifier, then classifies the sentiment of the retrieved tweets with the domain-adversarial neural network.

Chowdhury et al. (2019) proposed to extract disaster-related keyphrases for filtering relevant tweets to enhance situational awareness. They propose to improve the previous keyphrase extraction model by incorporating contextual word embeddings, POS-tags, phonetics, and phonological features. Chowdhury et al. (2020) constructed a unique dataset of disaster-related tweets annotated with hashtags that can be used to filter actionable tweets. They build LSTM models within a Multi-Task Learning framework for predicting the hashtags using this dataset. Desai et al. (2020) built an emotion dataset of 15,000 English tweets spanning three hurricanes: Harvey, Irma, and Maria. They present a comprehensive study of fine-grained emotions and propose classification tasks to discriminate between coarse-grained emotion groups. They use unlabeled Twitter data to further train the BERT model and achieve the best results overall when comparing to other models such as CNNs.

Khare (2020) investigated similar problems to this thesis from a semantics perspective. While this thesis also includes approaches for classifying information for different types of crisis, and also for different language, their approaches are based on semantic enrichment of data through entity linking and expansion of context via knowledge bases such as DBpedia and Wikipedia.

Ma (2019) applied BERT on crisis tweets classification for multi-class task. The report proposed customized BERT with neural network and CNN as well as LSTM as the last layer for classification with representations from BERT. The experiments are run on several datasets combined together ignoring the disaster differences. Some of those datasets are also

used in this thesis, and we use similar models in the multi-source setting and compare with MDAN from (Zhao et al., 2017). Fan et al. (2020) proposed a pipeline aiming to unfold how a disaster affects different areas using social media data. The authors use Hurricane Harvey as an example and showed that this pipeline can be used for automated mapping of an event across time and affected areas, and thus can be a useful tool to help different stakeholders during a disaster. A fine-tuned BERT model is used to classify posts to humanitarian categories (e.g., damages, rescue etc.) in the pipeline's learning module. Named Entity Recognition (NER) is also used for detecting locations, and graph-based clustering is also used to identify credible situational information in the learning module. The final outputs of the pipeline are timelines of credible tweets of different situational categories for each affected area and how social media attentions changed according to time at different areas showed in maps.

As works from our groups, in (Mazloom et al., 2018) and (Mazloom et al., 2019), we proposed a hybrid feature-instance adaptation approach based on matrix factorization and the k-nearest neighbors algorithm. The proposed hybrid adaptation approach first applies matrix factorization to reduce the dimensionality of the data, and then uses a k-nearest neighbors algorithm to select a subset of the source disaster data that is representative for the target disaster. The selected subset is subsequently used to learn accurate Naive Bayes classifiers for the target disaster. Experimental results show that the approach can significantly improve the performance as compared with a baseline Naive Bayes classifier. Other group members also applied domain adaptation on image classifications as well as tweet classification. Li et al. (2019) proposed domain adaptation approaches for identifying disaster damages in images, and Li and Caragea (2020b) proposed domain adaptation with reconstruction for disaster tweet classification. The reconstruction process contains an autoencoder that reconstructs the target data, while the source shares the encoder and whose reduced representation is used to learn a source classifier.

Chapter 4

Overview of datasets used

In this thesis, we mainly use three publicly available labeled datasets for training and evaluating models. For the task of filtering crisis related tweets, we use a dataset of tweets called CrisisLexT6 (Olteanu et al., 2014) and another dataset about incidents like fires or gunshots from (Schulz et al., 2017), denoted by 2CTweets. For the task of filtering informative tweets, we use the dataset called CrisisLexT26 (Olteanu et al., 2015). CrisisLexT6 and CrisisLexT26 are available from the CrisisLex project website[1]. We obtained the 2CTweets dataset directly from the authors. We will present a brief description of these datasets here, and give further statistics after data cleaning in the Chapters 5, 6 when the datasets are used for evaluation. Besides these datasets, we also collected tweets through the Twitter Streaming API during the 2017 Hurricane season, specifically tweets about Hurricane Harvey, Hurricane Irma, Hurricane Maria, and also Mexico Earthquake. The total corpus contains approximately 5.8 million tweets. Most of these tweets are crisis related, but there is also some inherent noise due to the fact that the streaming is keyword-based. This corpus is used as unlabeled data when training crisis specific word embeddings. Finally for future work, there are also multiple datasets from the CrisisNLP project website, for example CrisisMMD dataset (Alam et al., 2018b) [2], where tweets are labeled with situational awareness category labels such as infrastructure damage, volunteer and donation, etc.

[1] http://crisislex.org/data-collections.html
[2] https://crisisnlp.qcri.org/

4.1 Relevant vs. irrelevant task

4.1.1 CrisisLexT6

The tweets in this dataset are collected through Twitter API based on keywords and geo-locations of affected areas, and manually labeled as *on-topic* (i.e., relevant) or *off-topic* (i.e., irrelevant) to a disaster using the crowdsourcing platform CrowdFlower (currently, renamed FigureEight). As we have discussed in the introduction, for a new disaster, the task of identifying tweets relevant to that disaster (on-topic), among all the tweets posted during the disaster, is the first task that needs to be addressed. Furthermore, this task is particularly suitable for domain adaptation, which uses prior source labeled data together with target unlabeled data, and can thus capture specific patterns in the target data itself.

The CrisisLexT6 dataset contains six disasters occurring between October 2012 and July 2013 in USA, Canada and Australia. There are approximately 10,000 labeled tweets for each disaster. The statistics for the dataset are shown in Table 4.1, organized based on the time when each disaster happened.

Table 4.1: Statistics about the dataset CrisisLexT6

	Before Cleaning		
Crisis	On-topic	Off-topic	Total
2012_Sandy_Hurricane	6138	3870	10008
2013_Queensland_Floods	5414	4619	10033
2013_Boston_Bombings	5648	4364	10012
2013_West_Texas_Explosion	5246	4760	10006
2013_Oklahoma_Tornado	4827	5165	9992
2013_Alberta_Floods	5189	4842	10031

4.1.2 2CTweets

2CTweets, is a collection of tweets about incidents, such as car crash, fire or shooting, which happened in 10 different cities, as shown in Table 4.2. Tweets are labeled as incident related (*Yes*) or not (*No*). Given that incidents in different cities most likely involve local named

entities, such as local street names, adaptation is needed to enable generalization of classifiers between different cities (Schulz et al., 2017). The statistics for the dataset are shown in Table 4.2, organized roughly based on the time when tweets of each city were posted[3].

Table 4.2: Statistics about the dataset 2CTweets

	Before Cleaning		
2CTweets	Yes	No	Total
Memphis	361	721	1082
Seattle	800	1404	2204
NYC	413	1446	1859
Chicago	214	1270	1484
San Francisco	304	1176	1480
Boston	604	2216	2820
Brisbane	689	1898	2587
Dublin	199	2616	2815
London	552	2444	2996
Sydney	852	1991	2843

4.2 Informative vs. non-informative task

4.2.1 CrisisLexT26

CrisisLexT26, is a collection of tweets posted during twenty six crisis events that happened in 2012 or 2013, with most events having between 2,000 and 4,000 tweets. These tweets are also collected using filtering keywords, and labeled by CrowdFlower workers according to informativeness (i.e., *informative* or *non-informative*), information types (e.g., *caution and advice*, *infrastructure damage*), and information sources (e.g., *Governments*, *NGOs*). As we are focusing on English language in this thesis, we only selected seven events that mainly have English tweets, as shown in Table 4.3. For this dataset, we focus on task of classifying tweets as *informative* or *non-informative*. The statistics for the dataset are shown in Table 4.3, organized based on the time when each disaster happened.

[3]As some tweets of several cities are collected at the same time

Table 4.3: Statistics about the dataset CrisisLexT26

	Before Cleaning		
CrisisLexT26	Informative	Non-Informative.	Total
2012_Colorado_wildfires	685	268	953
2013_Queensland_floods	728	191	919
2013_Boston_bombings	417	512	929
2013_West_Texas_explosion	472	439	911
2013_Alberta_floods	685	298	983
2013_Colorado_floods	768	157	925
2013_NY_train_crash	904	95	999

Chapter 5

Self-labeling and correlation alignment approaches with Naive Bayes

In this chapter, we propose two domain adaptation approaches based on the supervised Naive Bayes classifier. The first proposed approach is a weighted Naive Bayes Classifier with either self-training strategy or Expectation-Maximization (EM) strategy for handling target unlabeled data. We compare the EM approach with self-training and find self-training gives better results. Following that, we perform an extensive study to select hyper-parameters for the weighted Naive Bayes classifier with self-training approach. The goal is to provide guidance on practical usage of this approach. The second proposed approach combines a feature-based domain adaptation approach called Correlation Alignment (CORAL) with the first approach. We compare the results between the two approaches, i.e., weighted Naive Bayes with self-training strategy and the Correlation Alignment extension. Both approaches have better performance than the corresponding base classifiers and are computationally efficient. In the following sections, we will first introduce the dataset used here, the cleaning steps and the setup of pairs of source and target domains in Section 5.2.2. Then we will present the first approach in detail, and discuss the experimental setup and results, followed

by the extensive experiments for parameter tuning in Section 5.2. Finally, we present the second approach, a hybrid model, and discuss the results in Section 5.3.

5.1 Naive Bayes

We have briefly introduced Naive Bayes classifiers in Chapter 2. We now present the details of the Naive Bayes classifiers used in this chapter. Our domain adaptation approaches in this chapter use the multivariate Bernoulli Naive Bayes (Manning et al., 2008) and Gaussian Naive Bayes model. Let's first look at Bernoulli Naive Bayes model. Given a collection of documents D as training set, each document $d_i \in D$, $(i = 1, \ldots, N)$, represents a data instance, and has a class label $c_k \in C$ associated with it. The set of words w_t in the collection D corresponds to the set of features used to represent documents, a.k.a., vocabulary V. Using the features in V, each document d_i is represented as a $|V|$ dimensional vector of *0*s and *1*s, based on the occurrence of word $w_t \in V$ in document d_i. Using the Bayes rule and the assumption that features are independent given the class, the class label for a new document d can be obtained as:

$$c^* = \underset{c_k}{\operatorname{argmax}}\, P(c_k|d) = \underset{c_k}{\operatorname{argmax}}\, \frac{P(d|c_k)P(c_k)}{P(d)} = \underset{c_k}{\operatorname{argmax}}\, P(c_k) \prod_{t=1}^{|V|} P(w_t|c_k) \tag{5.1}$$

Therefore, to be able to predict the class label for new documents d, we need to estimate the prior class probabilities $P(c_k)$ for all $c_k \in C$, and the likelihoods $P(w_t|c_k)$ for all $w_t \in V$ and $c_k \in C$. Estimation of the class priors and likelihoods can be done based on a training data. Specifically, we estimate the class priors and likelihoods from the training data, using the add-1 smoothing strategy (to avoid zero probabilities), as follows:

$$P(c_k) = \frac{N(c_k) + 1}{N + 1} \tag{5.2}$$

$$P(w_t = 0|c_k) = \frac{N(w_t = 0, c_k) + 1}{N(c_k) + 2} \tag{5.3}$$

$$P(w_t = 1|c_k) = \frac{N(w_t = 1, c_k) + 1}{N(c_k) + 2} \tag{5.4}$$

where N is the total number of documents in the collection D, $N(c_k)$ is the number of documents in class c_k, $N(w_t = 0, c_k)$ is the number of documents in class c_k that don't contain the word w_t, and $N(w_t = 1, c_k)$ is the number of documents in class c_k that contain the word w_t.

For Gaussian Naive Bayes, the estimation of priors is the same as in Equation 5.2. For likelihood estimation, the likelihood is assumed to be Gaussian, and is estimated with:

$$P(w_t|c_k) = \frac{1}{\sqrt{2\pi\delta_{tk}^2}} \exp - \left(\frac{(w_t - \mu_{tk})^2}{2\delta_{tk}^2} \right) \tag{5.5}$$

where μ_{tk} is the mean of feature t of class c_k, δ_{tk}^2 is corresponding variance.

5.2 Self-training/EM with Naive Bayes classifier

5.2.1 Self-training/EM with Naive Bayes

Here we present the Bernoulli Naive Bayes classifier with self-training, as well as with the iterative Expectation-Maximization (EM) approaches. We may implicitly refer to these two approach as Self-Training (ST) approach and EM approach in this chapter. In the EM approach, a classifier is learned at each iteration, and used to label the target unlabeled data. Subsequently, the target data, with probabilistic soft-labels assigned by the current classifier (e.g., $p(+|d) = 0.7$ and $p(-|d) = 0.3$ for an instance d), are combined with the labeled source data and used to train the classifier at the next iteration. The original classifier is trained from source data only. The process continues for a fixed number of iterations, or until convergence.

Similar to the EM approach, Self-Training approach is also an iterative approach that uses a weighted Naive Bayes classifier to combine source and target data. Same as the EM approach, it starts by learning a supervised classifier from source data only, and uses

that classifier to label the target unlabeled data. However, instead of adding all the target data with probabilistic soft-labels to the training set for the next iteration as in EM, in self-training, only the most confidently classified data are added to the training set, with hard (e.g., $+/-$ or *1/0*) labels. More precisely, only the most confidently labeled instances (e.g., the top k instances in each class based on the posterior class distribution, respectively) are added to the training set at subsequent iterations. And a new classifier is learned from the added target instances together with the original source instances. Given that we use the most confidently labeled target instances, which presumably have high posterior class distributions, we incorporate these instances with hard-labels, as opposed to probabilistic soft-labels, to help keep a cleaner decision boundary between the two classes.

The details of the algorithm with Self-Training approach are shown in Algorithm 1. With EM strategy, we just need to change Step 2 and the M-step and E-step. In step 2, for EM, we use all unlabeled target examples with soft labels instead of top-k examples with hard labels. And then we calculate weighted priors and likelihood and posteriors with soft labels. We denote the training source labeled data by tSL and the training target unlabeled data by tTU.

5.2.2 Data preprocessing and setup

We evaluate the proposed methods on CrisisLexT6 (Olteanu et al., 2014) for relevant vs. irrelevant task. We use the bag-of-words *0/1* representation to represent a tweet as a vector of features. We clean the tweets as follows:

1. We remove non-printable, ASCII characters, as they are generally regarded as noise rather than useful information.

2. We convert printable HTML entities into their corresponding ASCII equivalents.

3. We replace URLs, email addresses, and usernames with a URL/email/username placeholder for each type of entity, respectively.

4. We keep numbers, punctuation signs and hashtags, under the assumption that numbers

Algorithm 1: Pseudocode for Naive Bayes Classifier with Self-Training strategy

1. Simultaneously estimate the priors and likelihoods (a.k.a., train a Naive Bayes classifier) for the source domain:

$$P(c_k) = P_{tSL}(c_k)$$

$$P(w_t|c_k) = P_{tSL}(w_t|c_k)$$

2. Use the classifier learned from source to assign labels to the unlabeled instances from the target domain, and select the most confidently labeled instances (based on the prior class distribution, e.g., top 5 instances in each class, for a balanced dataset) as hard-labeled instances for self-training.

3. Loop until the labels assigned to the remaining unlabeled target instances don't change:

   (a) **M-step**: Same as Step 1, but use also the target instances labeled so far:

$$P(c_k) = (1-\gamma) \cdot P_{tSL}(c_k) + \gamma \cdot P_{tTU}(c_k)$$

$$P(w_t|c_k) = (1-\gamma) \cdot P_{tSL}(w_t|c_k) + \gamma \cdot P_{tTU}(w_t|c_k)$$

   where $\gamma = min(\tau\delta, 1)$, τ is the iteration number, and δ is a parameter that determines how fast we shift the weight from the source labeled data used for training (tSL) to the (originally unlabeled) target data used for training (tTU).

   (b) **E-step**: Calculate the posterior class distribution for the remaining set of unlabeled instances from the target domain.

4. Use the final classifier to label test target unlabeled instances $d = (w_1, \ldots w_{|V|})$:

$$c^* = \underset{c_k}{\operatorname{argmax}}\, P(c_k) \prod_{t=1}^{|V|} P(w_t|c_k)$$

   where V is the set of features/words used to represent instances.

could be indicative of an address, while punctuation/emoticons and hashtags could be indicative of emotions.

5. We remove RT (i.e., retweet), under the assumptions that such features are not informative for our classification tasks.

6. Finally, duplicate tweets and empty tweets (that have no characters left after the

cleaning) are removed from the data sets.

The statistics for the final dataset are shown in Table 5.1, organized based on the time when each disaster happened.

Table 5.1: Statistics for CrisisLexT6 before cleaning and after cleaning

	Before Cleaning			After Cleaning		
Crisis	On-topic	Off-topic	Total	On-topic	Off-topic	Total
2012_Sandy_Hurricane	6138	3870	10008	5261	3752	9013
2013_Queensland_Floods	5414	4619	10033	3236	4550	7786
2013_Boston_Bombings	5648	4364	10012	4441	4309	8750
2013_West_Texas_Explosion	5246	4760	10006	4123	4733	8856
2013_Oklahoma_Tornado	4827	5165	9992	3209	5049	8258
2013_Alberta_Floods	5189	4842	10031	3497	4714	8211

From Table 5.1, we can see that the CrisisLexT6 disaster datasets are fairly balanced (i.e., the ratio of on-topic to off-topic tweets is close to 1).

We follow the timeline of the six disasters in the dataset and select a variety of disaster pairs to perform experiments. Except for Hurricane Sandy, which does not have a prior disaster in this dataset, all the other disasters are used as target disasters in one or more pairs. We end up with 11 pairs, which cover natural disaster pairs, man-made disaster pairs, and also natural and man-made disaster pairs. The pairs also contain disasters of different types as well as pairs of disasters of the same type (e.g. Queensland Floods to Albert Floods). Intuitively, the similarity in terms of the type of the disaster should generally help. When reporting the results, we arrange pairs having the same target disaster but different source disasters together, and represent each pair with its initials of the source and target disasters, as shown in Table 5.2.

5.2.3 Experimental setup

Our goal is to evaluate the proposed domain adaptation approaches for the task of identifying tweets relevant to a target disaster (i.e., on-topic versus off-topic tweets). Our main working hypothesis is that the domain adaptation approach, which makes use of target unlabeled data in addition to source labeled data, can better capture patterns specific to the target

Table 5.2: Source-target pairs of disasters used in the experiments

Pair	Source Disaster	Target Disaster
SH → QF	2012_Sandy_Hurricane	2013_Queensland_Floods
SH → BB	2012_Sandy_Hurricane	2013_Boston_Bombings
QF → BB	2013_Queensland_Floods	2013_Boston_Bombings
SH → WTE	2012_Sandy_Hurricane	2013_West_Texas_Explosion
BB → WTE	2013_Boston_Bombings	2013_West_Texas_Explosion
SH → OT	2012_Sandy_Hurricane	2013_Oklahoma_Tornado
QF → OT	2013_Queensland_Floods	2013_Oklahoma_Tornado
BB → OT	2013_Boston_Bombings	2013_Oklahoma_Tornado
SH → AF	2012_Sandy_Hurricane	2013_Alberta_Floods
QF → AF	2013_Queensland_Floods	2013_Alberta_Floods
BB → AF	2013_Boston_Bombings	2013_Alberta_Floods

as compared to a supervised learning approach that would use only source labeled data. To verify this hypothesis, we ask the following questions:

- How do supervised classifiers learned only from source labeled data perform on target data?
- How do the results of the domain adaptation classifiers, which use both source labeled data and target unlabeled data, compare with the results of the supervised classifiers, which use only source data, when used to classify target data?

Given that the proposed domain adaptation approach can work with Self-Training with hard-labeled target data, as well as EM with soft-labeled target data, our next research question is:

- How do the results of the self-training strategy with hard-labeled target data compare with that of the EM strategy with soft-labeled target data?

Finally, we want to see how the results of the domain adaptation approach compare with the results of ideal supervised learning classifiers trained from target labeled data, with the assumption that time and effort would be spent to manually label the available unlabeled target data. Thus, our last research question is:s

- How close are the results of the domain adaptation classifiers to the results of supervised classifiers learned from a large amount of target labeled data?

For each pair of source and target disasters in Table 5.2, we use 5-fold cross-validation to select the target data to be used as unlabeled and test data. Specifically, the target data is split into five folds; at each rotation, one fold is selected as target test (TT), and three folds as target unlabeled data (tTU), used by the domain adaptation approach with self-training/Expectation-Maximization, together with source labeled data (tSL). The fifth fold is reserved as target labeled data to be used in future work.

We report the results using accuracy and area under the receiver operating characteristic curve ($auROC$) averaged over the five target test folds. Both accuracy and $auROC$ are metrics commonly used in machine learning, and capture different qualities of a classifier. The accuracy measures the percentage of correctly labeled instances out of the total number of instances. The $auROC$ measures the ability of the classifier to rank instances based on the probability, $P(c|d)$, that they belong to a class (positive or negative), without effectively assigning instances to classes. Given a ranking, the ROC plots the true positive rate as a function of the false positive rate, obtained at different cut-points in the ranking. As opposed to that, the accuracy is obtained based on one single cut-point (most commonly 0.5).

For each pair of source-target disasters, we perform four groups of experiments as described below, one for each of our research questions stated above.

Supervised learning from source labeled data only. In this group of experiments, we use source labeled data as the training set, and learn supervised Naive Bayes classifiers. We then use the resulting classifiers to classify target test data. Thus, the classifiers learned in this group of experiments can serve as baselines (intuitively, lower bounds) for the domain adaptation classifiers. We denote the supervised Naive Bayes classifiers by NB-S. The training data for this classifier, training source labeled data, is denoted by tSL. Given that other supervised classifiers have been used successfully in prior work, we also compare the results of the supervised Naive Bayes classifiers with the results of supervised random forest (RF), logistic regression (LR) and support vector machine (SVM) classifiers. One advantage of the

Naive Bayes classifier over other classifiers, and the reason our domain adaptation approach uses it as base classifier, is that the Naive Bayes algorithm does not have any parameters that require tuning. We used an open-source machine learning library, called WEKA (Hall et al., 2009), to learn supervised classifiers. We used default parameters for the RF, LR, SVM algorithms.

Domain adaptation with Self-Training and Expectation-Maximization, respectively. There are two groups of domain adaptation experiments. One is domain adaptation with self-training and hard-labeled target data, the other is domain adaptation with Expectation-Maximization and soft-labeled target data. In both groups, we use source labeled data and target unlabeled data to train a domain adaptation classifier for the target, and subsequently test the classifier on the target test data. We use the notation NB-EM for domain adaptation with the EM strategy, and NB-ST for domain adaptation with the Self-Training(ST) strategy. The training data for this classifier is denoted by $tSL+tTU$ to suggest that it is consists of source labeled data and target unlabeled data.

Supervised learning from target labeled data. In this group of experiments, we use the target unlabeled dataset (tTU) that is used in the domain adaptation setting and assume that the labels of the instances in this dataset are provided. We learn Naive Bayes classifiers from the target labeled data and test them on target test data. Intuitively, if labeled training data from a target disaster is available, we should be able to learn accurate classifiers for that disaster. Therefore, the results of the supervised classifiers learned from the assumed target labeled data can be seen as upper bounds for all the results of the other classifiers. We denote this supervised Naive Bayes classifiers by NB-T*.

Parameter tuning. The domain adaptation approach has two parameters that need to be tuned: the parameter δ that controls how fast we shift the weight from source to target in both ST and EM strategies; and the parameter k that controls how many instances to hard-label at each iteration of the ST strategy. To avoid over-fitting, we tune parameters during

a validation step. For the validation, we select one of three target unlabeled (tTU) folds as validation data (TV), and use the other two folds of tTU as target unlabeled data (tUV). We use *tSL+tUV* for training and test on TV to select the best values for the parameters. After tuning, the whole tTU is used for self-training as well as for EM. The performance metrics are estimated using the target test set TT. The following values are considered for the parameter δ: 0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9. In addition, in Self-Training approach, we hard-label k=1, 5, or 10 instances of on-topic and off-topic, respectively.

5.2.4 Results and discussion

Tables 5.3 and 5.4 show the results of the supervised classifiers learned from source data only in terms of accuracy and auROC (averaged over 5-folds), respectively. The results of the domain adaptation classifiers compared with supervised Naive Bayes classifiers are shown in Tables 5.5 and 5.6 in terms of accuracy and auROC, respectively.

For supervised models, we compare the following classifiers: Naive Bayes (NB-S), Support Vector Machines (SVM-S), Random Forests (RF-S) and Logistic Regression (LR-S). As mentioned earlier, we used the Weka implementations (Hall et al., 2009), with default parameters. Furthermore, we used the LibLinear variant of SVM (Fan et al., 2008), as opposed to LibSVM variant (Chang and Lin, 2011), as the results were consistently better for the LibLinear variant.

For domain adaptation approaches, we compare the baseline NB-S with Expectation-Maximization (NB-EM) and Self-Training (NB-ST), respectively as well as NB-T* which corresponds to an ideal classifier learned from target labeled data. The results of this classifier, underlined, can be seen as an upper bound for the results that can be achieved with domain adaptation which has access to only unlabeled data from the target domain, in addition to labeled data from a prior source domain.

We perform paired t-tests to identify classifiers that are statistically significantly better than their counterparts (using $p < 0.05$). The best values(not include NB-T*) for a pair are shown in bold. Furthermore, if a result is equivalent to the result of the ideal NB-

T* classifier, it is indicated with underscore, and underscore with star (*) means that the corresponding domain adaptation result is better than the result of the NB-T* classifier.

Table 5.3: Accuracy results of supervised classifiers trained from labeled source data only.

S → T	NB-S	SVM-S	RF-S	LR-S
SH → QF	**76.84**	73.81	70.11	71.85
SH → BB	68.66	55.23	**73.33**	56.41
QF → BB	**74.97**	65.76	71.65	58.00
SH → WTE	**77.21**	63.43	**77.16**	64.86
BB → WTE	**94.77**	84.29	92.15	87.85
SH → OT	**80.78**	77.15	**79.78**	72.6
QF → OT	**84.13**	**83.97**	81.56	79.01
BB → OT	**84.35**	81.45	82.45	79.78
SH → AF	**71.06**	68.86	65.85	67.09
QF → AF	**78.87**	76.69	74.49	72.06
BB → AF	**73.81**	71.17	**73.95**	66.42

Table 5.4: Weighted auROC results of supervised classifiers trained from labeled source data only.

S → T	NB-S	SVM-S	RF-S	LR-S
SH → QF	**0.911**	0.763	0.885	0.847
SH → BB	0.753	0.555	**0.825**	0.525
QF → BB	0.82	0.661	**0.833**	0.472
SH → WTE	0.853	0.626	**0.873**	0.598
BB → WTE	**0.983**	0.835	0.977	0.919
SH → OT	**0.865**	0.743	**0.860**	0.612
QF → OT	**0.880**	0.824	**0.899**	0.775
BB → OT	**0.905**	0.81	0.891	0.817
SH → AF	**0.830**	0.707	0.818	0.766
QF → AF	**0.860**	0.733	**0.860**	0.714
BB → AF	**0.806**	0.705	0.818	0.628

As can be seen from Tables 5.3 and 5.4, the Naive Bayes classifier has the overall best performance in terms of both accuracy and auROC metrics, when compared with other supervised classifiers trained with default parameters. Furthermore, the Naive Bayes classifier has the advantage that it does not require any parameter tuning. Given these reasons, we build our domain adaptation classifiers based on Naive Bayes, and compare the domain adaptation classifiers only with supervised Naive Bayes classifiers in what follows.

Table 5.5: Accuracy results of supervised classifiers (baseline and upperbound), EM and Self-training domain adaptation approaches

	NB-S	NB-EM	**NB-ST**	NB-T*
SH → QF	76.84	78.96	**82.40**	*93.42*
SH → BB	68.66	80.88	**84.06**	*89.20*
QF → BB	74.97	76.69	**81.86**	*89.20*
SH → WTE	77.21	**94.66**	90.82	*95.90*
BB → WTE	94.77	**95.79**	94.82	*95.90*
SH → OT	80.78	**87.58**	**87.76**	*90.45*
QF → OT	84.13	**86.63**	85.48	*90.45*
BB → OT	84.35	**87.44**	**86.91**	*90.45*
SH → AF	71.06	76.87	**82.57**	*92.63*
QF → AF	78.87	82.43	**86.01**	*92.63*
BB → AF	73.81	**82.47**	**83.96**	*92.63*

Table 5.5 shows the 5-fold average accuracy for each classifier and each pair. Table 5.6 shows the 5-fold average weighted auROC for each classifier and each pair. The first row in each of the result tables corresponds to the supervised Naive Bayes classifiers learned from source only (NB-S), the next two rows correspond to the domain adaptation approaches with Expectation-Maximization (NB-EM) and Self-Training (NB-ST), respectively. The last row in each table NB-T* corresponds to an ideal classifier learned from target labeled data. The results of this classifier, underlined, can be seen as an upper bound for the results that can be achieved with domain adaptation which has access to only unlabeled data from the target domain, in addition to labeled data from a prior source domain.

Based on the results in Tables 5.3, 5.4, 5.5 and 5.6, we answer our research questions below.

How do the supervised classifiers learned only from source labeled data perform on target data? As our results in Tables 5.3 and 5.4 show, labeled data from a prior source disaster can be very useful for learning classifier for different target disasters. When using only source labeled data to learn Naive Bayes classifier (the approach NB-S), the auROC values are greater than 0.8 or 0.9 for most pairs, with the exception of pair SH → QF, for which the auROC value is around 0.75. Similarly, the accuracy for most pairs is over 70% or

Table 5.6: Weighted auROC results of supervised classifiers (baseline and upperbound), EM and Self-training domain adaptation approaches.

S → T	NB-S	NB-EM	**NB-ST**	*NB-T**
SH → QF	0.911	**0.973***	**0.974***	*0.969*
SH → BB	0.753	0.929	**0.941**	*0.954*
QF → BB	0.82	0.832	**0.890**	*0.954*
SH → WTE	0.853	**0.984**	**0.987**	*0.989*
BB → WTE	0.983	**0.989**	0.984	*0.989*
SH → OT	0.865	**0.938**	**0.951**	*0.961*
QF → OT	0.88	**0.925**	**0.924**	*0.961*
BB → OT	0.905	**0.942**	**0.944**	*0.961*
SH → AF	0.83	0.953	**0.972**	*0.971*
QF → AF	0.86	0.882	**0.922**	*0.971*
BB → AF	0.806	0.898	**0.950**	*0.971*

80% except for pair SH → BB as well. The accuracy and auROC values are especially high when considering disasters of the same type (e.g., pair BB → WTE: Boston_Bombings → West_Texas_Explosion), and relatively smaller for more different disasters (e.g., pairs QF → BB: Quessland_Floods → Boston_Bombings, and BB → AF: Boston_Bombings → Alberta_Floods). Furthermore, it is worth noting that, while both pairs SH → AF and QF → AF have Alberta Floods as target, the results for QF → AF, which has Queensland Floods as source (another flood) are better than the results for SH → AF, which has Sandy Hurricane as source. Similar behavior is observed for pairs SH → WTE and BB → WTE, which both have West Texas Explosion as target: Boston Bombings as source in BB → WTE gives better results than Sandy Hurricane in SH → WTE. Together, these results show that supervised learning based on source can be used to learn classifiers for a target if the source and target disasters are similar. This conclusion is consistent with other prior studies (Imran et al., 2013b; Li et al., 2015; Verma et al., 2011).

A more interesting observation is that for the pair BB → WT, the supervised Naive Bayes classifier is highly accurate, with accuracy close to 95% and auROC close to 1.0. By examining sample tweets from the two disasters, we find that they share more common features than other pairs of disasters in our experiments. Reasons for the common features

include the fact that the West Texas Explosion happened shortly after the Boston Bombings, both disasters happened in US, and they were man-made. Thus, people who tweeted about West Texas Explosion often mentioned Boston Bombings as well. However, this is not the case for the pair QF → AF (Queensland Floods → Alberta Floods), where both the source and the target disasters are floods (natural disasters), with different geo-locations, but people don't talk about Alberta Floods in relation to Queensland Floods.

Another interesting observation can be made for pairs SH → BB and QF → BB that have Boston Bombings as target, but Hurricane Sandy versus Queensland Floods as sources, respectively. As Hurricane Sandy mostly affected the east coast of the US, one might expect that the classifier for pair SH → BB may give better accuracy than the classifier for pair QF → BB. However, this is not the case as can be seen in Table 5.3, which suggests that domain similarity or closeness of disasters can be more sensitive to the occurring times of the disasters rather than geo-locations, or other facts about the disaster types. More datasets and experiments are needed to get a firmer conclusion in this respect.

How do the results of the domain adaptation classifiers that use both source labeled data and target unlabeled data compared with the results of the supervised classifiers that use only source data, when used to classify target data? As can be seen from Tables 5.5 and 5.6, domain adaptation approaches that make use of target unlabeled data definitely help to improve the results of the classifiers learned from source data only. By comparing the results of the supervised NB-S with the results of the domain adaptation approaches NB-EM and NB-ST, we can see that all pairs of disasters considered except for BB → WTE, domain adaptation classifiers with either EM or ST are significantly better than the corresponding supervised classifiers. For some pairs, the improvement is very big; for example, for pairs SH → BB, SH → WTE, SH → AF and BB → AF, the accuracy in Table 5.5 has improved by more than 10% when using the domain adaptation approach NB-ST as compared to the supervised learning algorithm with source only. For pair BB → WTE, domain adaptation NB-EM with soft-labels still improves the accuracy, whereas domain adaptation NB-ST with hard-labels doesn't help much. The reasons for this may lie in the fact that the source itself is close to that target, and the instances added with self-training are not very different from

the source instances. Still the domain adaptation with EM give results that are better than the results of the supervised classifier, according to the t-test (at $p < 0.05$).

How do the results of the self-training strategy with hard-labeled data compare with those of the EM strategy with soft-labeled target data? When comparing the NB-ST approach (with hard-labels) with the NB-EM approach (with soft-labels), we can see that in general NB-ST performs better than NB-EM. More specifically, for 8 out of 11 pairs, NB-ST is either equivalent (3 pairs) or better (5 pairs) than NB-EM in terms of accuracy, and for all pairs except P11, either equivalent or better in terms of auROC. For pair P3, although the accuracy in Table 5.5 is higher for NB-EM, the weighted auROC in Table 5.6 is equivalent to NB-ST. NB-EM with soft-labels is statistically better than NB-ST with hard-labels only on pairs SH → WTE, QF → OT and BB → WTE. EM which is using all target unlabeled data at each iteration may provide more information than ST as the sources in pair SH → WTE, QF → OT are not only of different types as compared to the target, but also far in time, and thus the original classifier learned from them is not reliable enough to accurately label a small number of instances for the NB-ST approach. However, overall, we can confidently say that ST performs better than EM for our classification task.

How close are the results of the domain adaptation classifiers to the results of supervised classifiers learned from a large amount of target labeled data? Finally, to answer our last research question, from Tables 5.5 and 5.6, we can see that domain adaptation approaches can achieve results very close to the upper bound in several cases but not always. By comparing the accuracy results of EM/ST in Table 5.5 with the accuracy results obtained with the ideal NB-T* approach (used as an upper bound), we can see that the domain adaptation algorithms get close to the upper bound in a few cases, for example, for pairs SH → WTE, SH → OT, SH → AF and especially pair BB → WTE. However, in general, there is still significant room for improving the accuracy results of the domain adaptation classifiers. By comparing the auROC results in Table 5.6, we can see that the results of the domain adaptation classifiers are, in general, closer to the upper bounds except for pairs QF → BB, QF → OT. Furthermore, in some cases the results are even better than the upper bound, for example for pair SH → QF. This can be explained by the fact that the source

itself provides accurate results (auROC value higher than 0.9), which makes it possible to accurately label the originally unlabeled target data. Thus, the accurately labeled target data together with the source data produce classifiers that are better than those learned from labeled target data alone (which could be noisy).

5.2.5 Hyper-parameter analysis for practical usage

We have shown that the proposed Naive Bayes with self-training domain adaptation approach is effective in disaster related tweets classification task. Here, we want to provide some guidance on the use of the approach, from a practical point of view. First, we want to understand how many source instances are needed to learn an accurate classifier for a target disaster in a domain adaptation setting. Second, we aim to understand how the performance of the domain adaptation classifier varies with parameters δ and k values, and to select good overall values to use in practice. Third, we aim to study how the performance varies with different numbers of iterations, and to identify an appropriate number of iterations for good performance. More specifically, we ask the following questions:

- How many source labeled instances are needed to build an accurate classifier for the target?
- What values should we use in practice for the parameters δ and k of the domain adaptation algorithms?
- How many iterations are needed to build an accurate classifier for the target?

We run an extensive set of experiments to answer the above questions.

Experimental setup

Training and test data: For each pair, we use the same 5 folds splits of the target data. To choose different amounts of source labeled data for each pair, we randomly select 250, 500, 1000, 2000 instances from each class (on-topic/off-topic), and then finally include all

instances from each class. Thus, we end up with SL-500, SL-1000, SL-2000, SL-4000 and SL-ALL number of source instances, respectively. The number of all instances for each disaster is around 8000 after cleaning, except for Hurricane Sandy whose number is around 9000. To report best performance for a pair, we tune parameters δ and k during a validation step. Recall from subsection 5.2.3 that, we randomly select one of the three target unlabeled (tTU) folds as validation data (TTV), and use the other two folds of tTU as target unlabeled data for validation (TUV). We use tSL+TUV for training and TTV and select the best values for the parameters based on TTV. After tuning, the whole tTU is used to learn the final classifier for the target, and performance is estimated using the target test set TT. The values used for δ are: {0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}, and the values used for k are {1,5,10}.

Experiments: We perform several experiments on each pair of disasters. The first experiment is domain adaptation with self-training running until convergence, with parameter values tuned based on the validation data. We refer to this experiment as NB-STT-Conv. The second experiment, NB-STF-Conv, is similar to the first one, except that the algorithm runs with fixed values for parameters δ and k. We compare the results of these experiments to understand how much is lost, if anything, by fixing parameters.

To see whether the performance can still improve after convergence, we also vary the number of iterations for fixed δ and k parameters, beyond convergence. The following value are considered for the number of iterations: {10, 50, 100, 150, 200, 250, 300}, and the best number of iterations, among those considered, is identified. We refer to the experiment where the algorithm runs with fixed parameters δ and k, and fixed number of iterations as NB-STF-Iter. The results of all experiments are reported in terms of the area under the ROC curve (auROC), but other measures (e.g., accuracy) show similar trends.

Results and discussion

To answer our research questions, we first run the algorithm using source datasets of different sizes and tune parameters. We then analyze how the performance varies with different values

for parameters δ and k, when running until convergence. We identify the general best values for those parameters. Then, using the selected δ and k values, we study the performance of the algorithm when increasing the number of iterations, and identify a good number of iterations to use instead of convergence. We answer the research questions in the following discussion.

How many source labeled instances are needed to build an accurate classifier for the target?

Table 5.7 shows the variation of the performance with the size of the source dataset. We performed column-wise paired t-tests $p \leq 0.05$) to compare the results in a row and identify values that are significantly better than their counterparts. The best values for each row/pair are shown in bold. As can be seen, the performance generally increases with the amount of labeled source data. However, between using 4000 instances and using all instances (approximately 8000), the performance does not increase much. In fact, for 7 out of 11 pairs, the results obtained with 4000 instances are as good as the results obtained with all source instances, suggesting that the effort that goes into data labeling is not worth beyond 4000 instances. Furthermore, we can also see that as few as 500 source instances can produce classifiers with performance close to 90%, and most of the time better than the performance of the supervised classifiers learned from all the source data. Therefore, if labeling 4000 instances is not possible, the domain adaptation algorithm can still help as compared to the supervised learning algorithm.

What values should we use in practice for the parameters δ and k of the domain adaptation algorithms?

Figures 5.1 and 5.2 show the variation of performance with parameters δ and k, respectively. In each figure, the variation of performance when 500 source instances are used is shown on the left, whereas variation when 4000 source instances are used is shown on the right. We focus on 500 and 4000 instances, respectively, to understand if the best values for parameters are different for smaller versus larger source datasets. In all cases, the algorithm is run to convergence.

Subplots (a), (b), (c) in Figure 5.1, show the variation of the performance with δ when 500 source instances are used, and k is fixed to $1, 5, 10$, respectively. Similarly, subplots

Table 5.7: Variation of the performance of the domain adaptation algorithm with the size of the source dataset. The supervised Naive Bayes classifier learned from all source data is used as a baseline. Performance is reported as weighted auROC values obtained using parameter tuning (averaged over 5 folds). The algorithm terminates upon convergence (NB-STT-Conv). The best value in each row is shown in bold (based on a t-test with $p \leq 0.05$).

Pair	Baseline	SL-500	SL-1000	SL-2000	SL-4000	SL-All
SH → QF	0.911	0.945	0.955	0.964	0.971	**0.974**
SH → BB	0.753	0.877	0.914	0.926	0.935	**0.941**
QF → BB	0.820	0.840	0.863	0.866	**0.890**	**0.890**
SH → WT	0.853	0.973	0.972	0.976	**0.986**	**0.987**
BB → WT	0.983	0.969	0.972	0.972	**0.980**	**0.984**
SH → OT	0.865	0.921	0.932	0.944	**0.953**	**0.951**
QF → OT	0.880	**0.916**	**0.921**	**0.919**	**0.926**	**0.924**
BB → OT	0.905	0.919	0.924	0.927	**0.942**	**0.944**
SH → AF	0.830	0.925	0.942	0.956	**0.970**	**0.972**
QF → AF	0.860	0.880	0.890	0.892	0.918	**0.922**
BB → AF	0.806	0.898	0.912	0.935	**0.950**	**0.950**

(d), (e), (f) in Figure 5.1, show the variation of the performance with δ when 4000 source instances are used, and k is fixed to $1, 5, 10$, respectively. We can see that when we have 4000 source instances, the best results are generally obtained for a very small value of δ, specifically 0.001, regardless of the value used for k. This result suggests that a source with 4000 instances produces a reasonably good classifier in the first place, and therefore the shift from the source to the target should be done slowly to allow for the accumulation of accurately labeled target instances in the training set. Another interesting observation is that for values of δ greater than 0.1, the performance does not change much. This is because when δ is large, the algorithm shifts all the weight to the target data in a small number of iterations. For example, when $\delta = 0.2$, the weight assigned to the target will be 1.0 at the sixth iteration ($\min(5 * 0.2, 1) = 1$), and thus the classifier solely depends on the self-training of the target unlabeled instances added in the first 5 iterations, which will lead the algorithm to converge very fast.

When 500 source instances are used, the best results are also obtained with small values

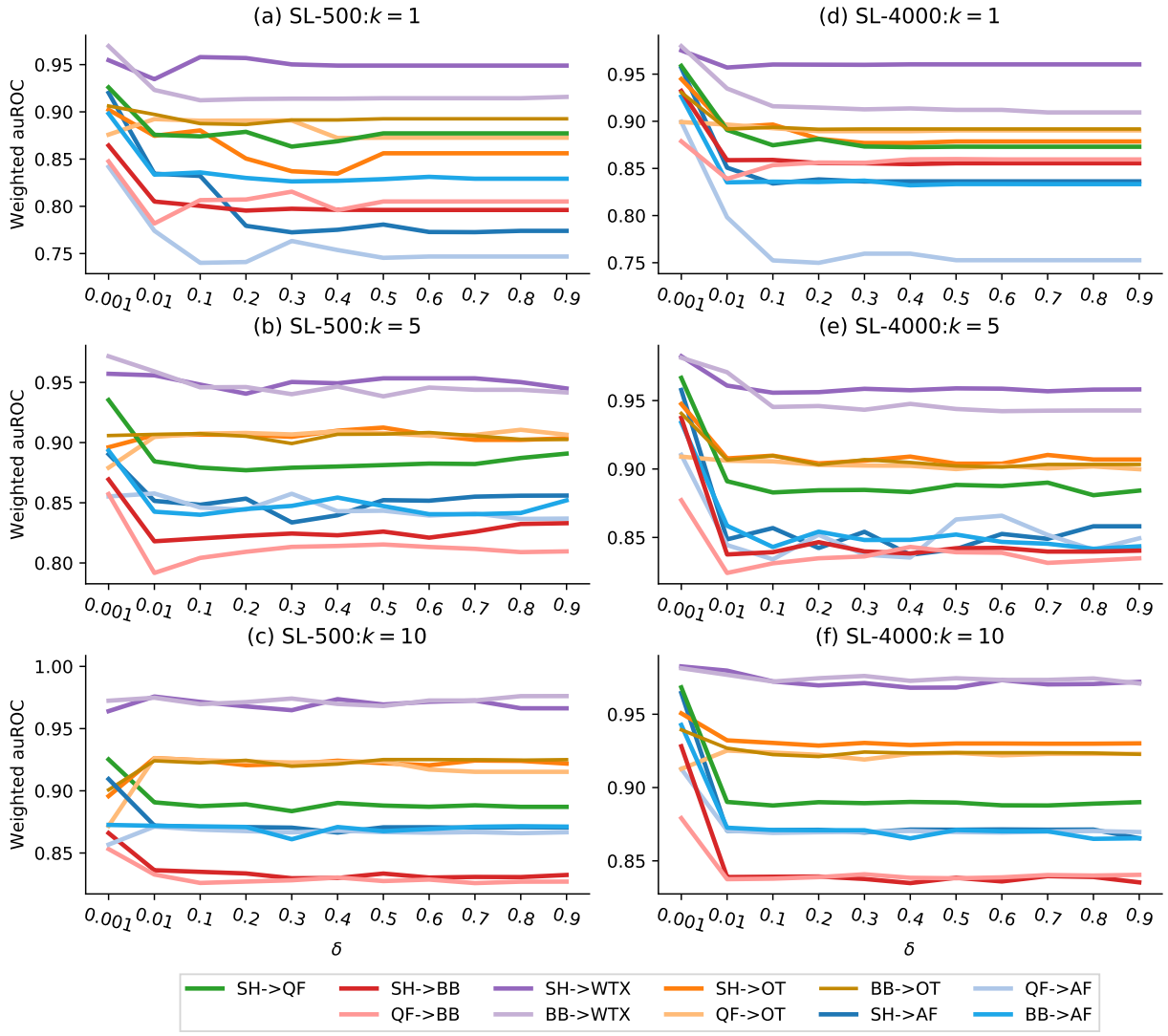

Figure 5.1: Variation of the performance (auROC) with δ for two sizes of the source dataset: SL-500 (left) and SL-4000 (right). The parameter k is fixed to 1, 5, and 10, respectively. The algorithm terminates upon convergence.

of δ, but the best value is not consistent as 0.001. In some cases, the best δ value is 0.01 or 0.1, which shows that the shift from source to target happens faster when the original classifier learned from source is not very good. In effect, a higher weight will be assigned to the target data, which is still small in size and possibly not very accurate. Given that, from a practical point of view, it is desirable to have a larger amount of source data (approximately 4000), as that makes it easier to find good overall values for the parameter δ.

Subplots (a), (b), (c) in Figure 5.2 show the variation of the performance with k when 500

source instances are used, and δ is fixed to $0.001, 0.01, 0.1$, respectively. Similarly, subplots (d), (e), (f) in Figure 5.2 show the variation of the performance with δ when 4000 source instances are used, and δ is fixed to $0.001, 0.01, 0.1$, respectively. By analyzing these plots, we can see that when δ is very small, for example 0.001, the performance is more steady, generally increasing very slowly with k, with some exceptions (e.g., SH $\rightarrow$ AF in Figure 5.2 (a)). In particular, for 4000 source instances and $\delta = 0.001$, the best performance is observed for either $k = 5$ or $k = 10$. The performance decrement from $k = 5$ to $k = 10$ that is observed for some pairs can be explained by the addition of mislabeled target instances to the training data, which can easily happen when too many target instances are added at once. When δ is 0.01 or 0.1, the increase/decrease pattern is less consistent overall, although for many pairs larger k is better.

Figures 5.1 and 5.2 together suggest that the best performance overall is obtained when the source data consists of approximately 4000 instances, parameter δ is set to 0.001 and parameter k is set to 5 or 10. Furthermore, even when only 500 source instance are available, the same parameters can be used.

To understand if performance is sacrificed when fixing parameters as opposed to tuning them, we compare the two settings when the algorithm runs to convergence. The results are shown in Table 5.8. As can be seen, the results obtained using fixed parameter values are as good as the results obtained using tuned values for most pairs, with only two exceptions for ST-4000 (QF $\rightarrow$ BB, QF $\rightarrow$ OT) and two exceptions for ST-500 (QF $\rightarrow$ OT and SH $\rightarrow$ WTE), where the performance with fixed values is just slightly worse than the performance with tuned values.

How many iterations are needed to build an accurate classifier for the target?

Using the findings about best overall parameters, our next objective is to compare the performance when the algorithm terminates upon convergence versus performance when the algorithm terminates after a fixed number of iterations, identified as a good overall number of iterations during validation. Specifically, we run experiments with fixed parameters δ and k, and vary the number of iterations. The results on the target validation data are shown in Figure 5.3, for 500 source instances (left) and 4000 source instances (right). The dot on each

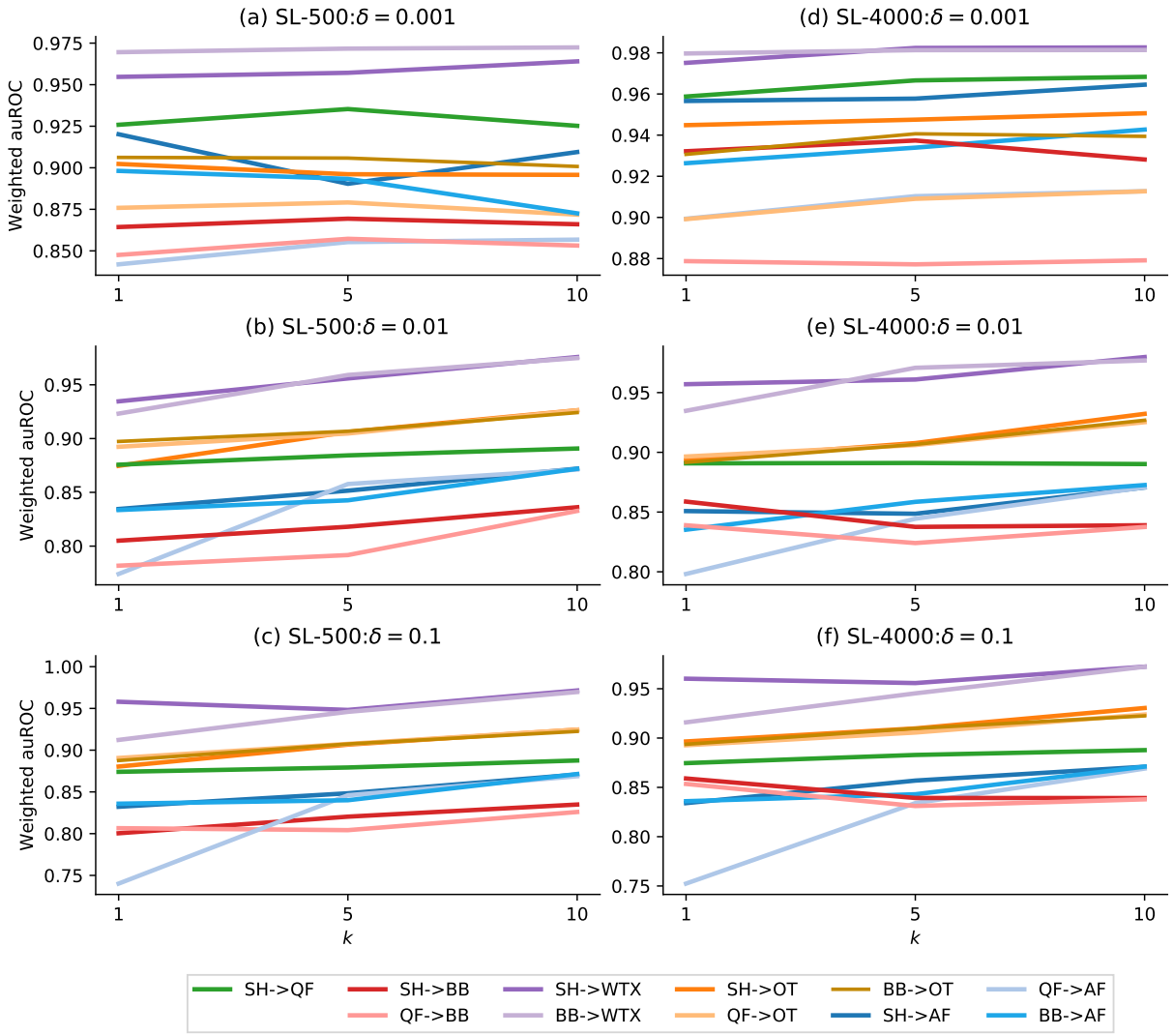

Figure 5.2: Variation of performance (auROC) with k for two sizes of the labeled source data: SL-500 (left) and SL-4000 (right). Parameter δ is fixed to 0.001, 0.01 and 0.1, respectively. The algorithm terminates upon convergence.

curve represents the number of iterations at "convergence" for the corresponding pair on a curve. As can be seen, the performance generally increases with the number of iterations, beyond the number of iterations at which "convergence" is reached. However, after a certain number of iterations (equivalently, after a certain number of target instances have been added to the training set), the performance does not change much. As expected, the number of iterations at which performance stabilizes is larger for $k = 5$ as compared to $k = 10$, as less target instances are added to the training data, at each iteration, for $k = 5$. In particular, the

Table 5.8: Tuning results (NB-STT-Conv) versus results with fixed parameters $\delta = 0.001$ and $k = 5$ (NB-STF-Conv). In both cases, the algorithm terminates upon convergence. Best value in each column of a pair is shown in bold (row-wise t-test with $p \leq 0.05$).

Pair	Experiment	SL-500	SL-4000
SH → QF	NB-STT-Conv	**0.945**	**0.971**
	NB-STF-Conv	**0.949**	**0.967**
SH → BB	NB-STT-Conv	**0.877**	**0.935**
	NB-STF-Conv	**0.873**	**0.938**
QF → BB	NB-STT-Conv	**0.840**	**0.890**
	NB-STF-Conv	**0.855**	0.878
SH → WTE	NB-STT-Conv	**0.973**	**0.986**
	NB-STF-Conv	0.959	**0.985**
BB → WTE	NB-STT-Conv	**0.969**	**0.980**
	NB-STF-Conv	**0.973**	**0.983**
SH → OT	NB-STT-Conv	**0.921**	**0.953**
	NB-STF-Conv	**0.910**	**0.931**
QF → OT	NB-STT-Conv	**0.916**	**0.926**
	NB-STF-Conv	0.893	0.914
BB → OT	NB-STT-Conv	**0.919**	**0.942**
	NB-STF-Conv	**0.911**	**0.942**
SH → AF	NB-STT-Conv	**0.925**	**0.970**
	NB-STF-Conv	**0.928**	**0.965**
QF → AF	NB-STT-Conv	**0.880**	**0.918**
	NB-STF-Conv	**0.864**	**0.919**
BB → AF	NB-STT-Conv	**0.898**	**0.950**
	NB-STF-Conv	**0.894**	**0.946**

performance becomes stable around 250/300 iterations for $k = 5$ and around 150 iterations for $k = 10$. For $k = 5$, 300 iterations correspond to 5x2x300=3000 target instances being added to the training data, while for $k = 10$, 150 iterations correspond to 2x10x150=3000 target instances as well. This result suggests that the number of target instances to be included in the training dataset needs to be greater than 3000 for best performance. However, it is important to note that the algorithm can start with the unlabeled target data available at the onset of a disaster. As more unlabeled target data becomes available at a later time,

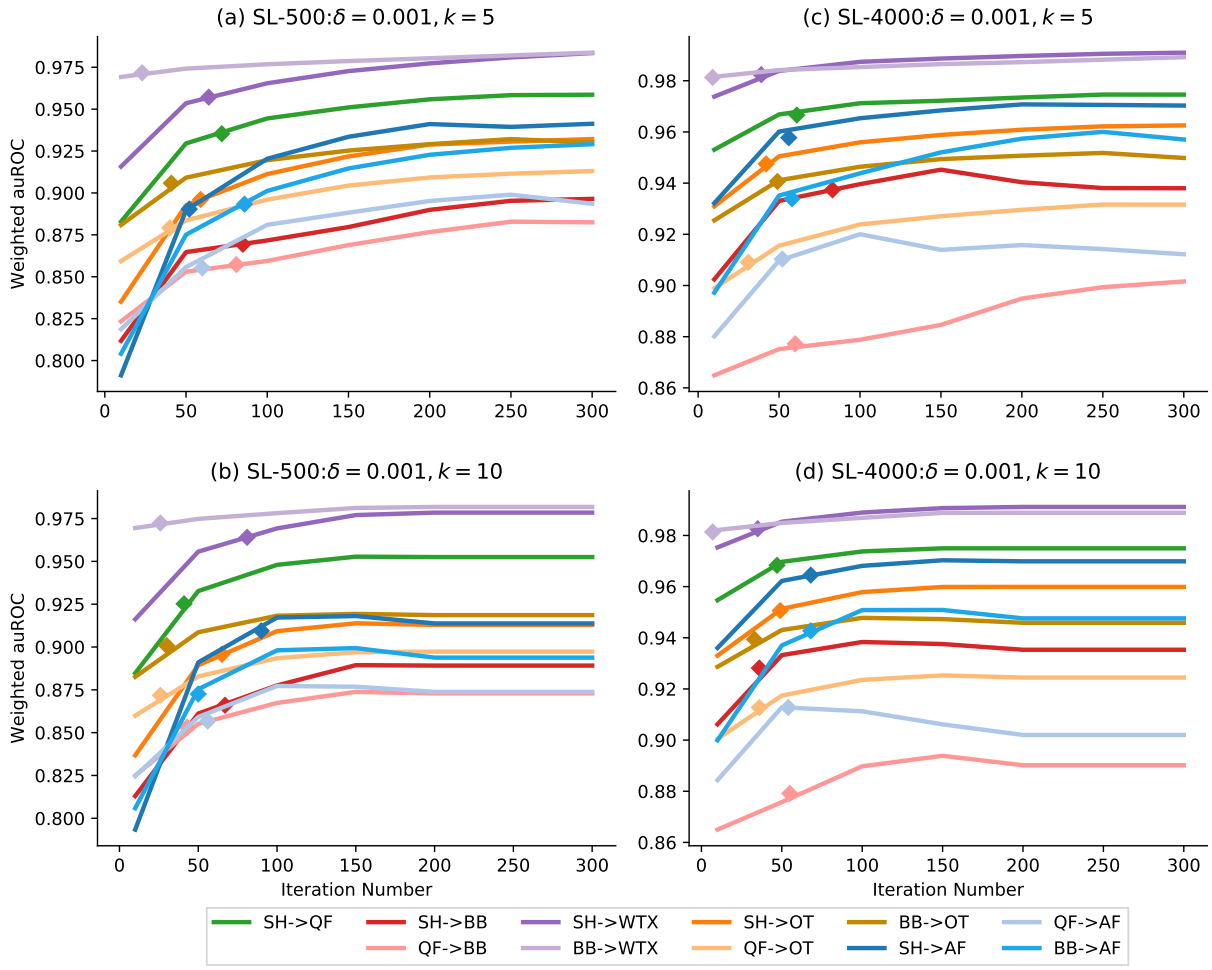

Figure 5.3: Variation of performance (auROC) with the number of iterations for two sizes of the labeled source data: SL-500 (left) and SL-4000 (right). The dots on each curve represent the performance at "convergence."

that data can be used in subsequent iterations of the domain adaptation algorithm, in an online fashion.

Table 5.9 shows the results of the algorithm when run with fixed parameters $\delta = 0.001$ and $k = 5$ and two termination conditions, respectively: convergence (NB-STF-Conv) and fixed number of iterations, specifically 300 iterations (NB-STF-Iter). As can be seen, the results are almost aways better when using a fixed number of iterations, therefore running the algorithm beyond pseudo-convergence is generally advantageous.

In summary, our empirical results suggest that fixing the algorithm's parameters δ and k, and fixing the number of iterations τ can be done without scarifying performance. In

Table 5.9: Results with fixed parameters when the algorithm runs to convergence (NB-STF-Conv) or for a fixed number of iterations (NB-STF-Iter), specifically 300 iterations. Best value in each column of a each pair is shown in bold (row-wise t-test with $p \leq 0.05$).

Pair	Experiment	SL-500	SL-4000
SH → QF	NB-STF-Conv	0.949	**0.967**
	NB-STF-Iter	**0.957**	**0.972**
SH → BB	NB-STF-Conv	0.873	**0.938**
	NB-STF-Iter	**0.893**	0.935
QF → BB	NB-STF-Conv	0.855	0.878
	NB-STF-Iter	**0.880**	**0.898**
SH → WTE	NB-STF-Conv	0.959	0.985
	NB-STF-Iter	**0.980**	**0.990**
BB → WTE	NB-STF-Conv	0.973	0.983
	NB-STF-Iter	**0.981**	**0.987**
SH → OT	NB-STF-Conv	0.910	0.950
	NB-STF-Iter	**0.938**	**0.962**
QF → OT	NB-STF-Conv	0.893	0.914
	NB-STF-Iter	**0.919**	**0.933**
BB → OT	NB-STF-Conv	0.911	0.942
	NB-STF-Iter	**0.934**	**0.952**
SH → AF	NB-STF-Conv	0.928	**0.965**
	NB-STF-Iter	**0.940**	**0.968**
QF → AF	NB-STF-Conv	0.864	**0.919**
	NB-STF-Iter	**0.902**	**0.916**
BB → AF	NB-STF-Conv	0.894	0.946
	NB-STF-Iter	**0.932**	**0.959**

turn, this finding makes it possible to use our domain adaptation approach in a practical situation, where target labeled data for tuning parameters is not available, but batches of unlabeled target data accumulate quickly in an online fashion.

5.3 Hybrid model combining Self-training and CORAL

Correlation alignment algorithm (CORrelation ALignment, CORAL) (Sun et al., 2015), is a feature-based adaptation approach that aligns the distribution of the source domain with

the distribution of the target domain to reduce the variance shift. Previous work on using CORAL (Sopova, 2017) to identify disaster relevant tweets has outperformed the Naive Bayes baseline. In this section, we will first compare CORAL with self-training to understand which approach benefits more from source adaptation based on unlabeled data. We will then design a hybrid approach that combines CORAL with self-training, with the goal of improving the results obtained with each independent approach further.

5.3.1 Correlation alignment

Introduced by Sun et al. (2015), CORrelation ALignment (CORAL) is a simple yet effective domain adaptation method. CORAL works by aligning the distributions of the source and target data in an unsupervised manner. More specifically, CORAL minimizes the domain shift by aligning the second-order statistics of source and target distributions, namely, the covariance, without requiring any target labels. As stated by Sun et al. (2015), CORAL aligns the distributions by re-coloring whitened source features with the covariance of the target features. The approach involves the following steps:

1. Compute covariance statistics in each domain, and

2. Apply the whitening and re-coloring linear transformations to the source features.

Then, supervised learning proceeds as usual – a classifier is trained using the transformed source features and used to classify the target data. Since the correlation alignment algorithm changes the features only, it can be applied to any base classifier.

Formally, we are given $D_S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{m_S}, y_{m_S})\} \subseteq \mathcal{X} \times \mathcal{Y}$ and target unlabeled data be $D_T = \{\mathbf{x}_1, \ldots, \mathbf{x}_{m_{TU}}\} \subseteq \mathcal{X}$, for easier discussion, let's first ignore the labels in D_S. Suppose μ_S, μ_T and C_S, C_T are the feature vector means and covariance matrices for source $\mathcal{D}_S$ and target $\mathcal{D}_T$, respectively. According to Sun et al. (2015), to minimize the distance between the second-order statistics (covariance) of the source and target features, one can apply a linear transformation A to the original source features. This transformation can be

obtain as a solution to the following minimization problem:

$$\min_A \|C_{\hat{S}} - C_T\|_F^2 = \min_A \left\|A^T C_S A - C_T\right\|_F^2, \tag{5.6}$$

where $C_{\hat{S}}$ is the covariance of the transformed source features $D_S A$, and $\|\cdot\|_F^2$ denotes the Frobenius norm of a matrix, used as a distance metric.

Let $D_S = U_S \Sigma_S V_S^\top$ be the singular value decomposition (SVD) of the matrix corresponding the the source data D_S, and similarly, let $D_T = U_T \Sigma_T V_T^\top$ be the SVD decomposition of the matrix corresponding the target data D_T. Furthermore, let $\Sigma_{T[1:r]}, U_{T[1:r]}, V_{T[1:r]}$ be the largest r singular values and the corresponding left and right singular vectors of D_T, respectively. As proven by Sun et al. (2015), the optimal solution for the above minimization problem can be found as:

$$A^* = \left(U_S \Sigma_S^{+\frac{1}{2}} U_S^\top\right) \left(U_{T[1:r]} \Sigma_{T[1:r]}^{+\frac{1}{2}} U_{T[1:r]}^\top\right), \tag{5.7}$$

which can be interpreted as follows: the first part is used to whiten the source data, while the second part is used to re-color it with the target covariance.

As Sun et al. (2015) suggest, after CORAL transforms the source features according to the target space, a classifier $h(\mathbf{x}; \mathbf{w})$ parametrized by $\mathbf{w}$ can be trained on the adjusted source features and directly applied to target features. In this work, we run experiments using Naive Bayes Classifier. The CORAL algorithm is summarized in Algorithm 2.

5.3.2 Experimental setup

We will use the same dataset as shown in subsection 5.2.2 for evaluation here.

Feature Selection

Sun et al. (2015) used a sentiment analysis dataset, where the dimensionality was reduced based on the information gain criterion. Sopova (2017) uses a different criterion for feature selection, specifically an unsupervised feature selection algorithm called "Variance Thresh-

Algorithm 2: Correlation Alignment Algorithm

Input: Target unlabeled data D_T, source labeled data D_S

Output: Adjusted source labeled data D_{S^*}

import numpy as np

from scipy.linalg import fractional_matrix_power

$ncolsS = D_S.shape[1]$

$ncolsT = D_T.shape[1]$

$C_S = np.cov(D_S, rowvar = 0) + np.eye(ncolsS)$

$C_T = np.cov(D_T, rowvar = 0) + np.eye(ncolsT)$

$D_S = np.dot(D_S, fractional\_matrix\_power(C_S, -0.5))$

$D_{S^*} = np.dot(D_S, fractional\_matrix\_power(C_T, 0.5))$

old". To compare, we will also use this feature selection technique in the hybrid model. Essentially, we remove all lowvariance features from the target data. The low-variance features are defined as those that are either 0 or 1 in more than $k\%$ of the samples, which corresponds to the Variance Threshold equal to $0.k * (1 - 0.k)$. The Variance Threshold looks only at the features X, but not at the class labels y. In order to select features, we first concatenate labeled source data D_S and unlabeled target data D_T. Once a subset of the features is selected, we represent D_S and D_T using the selected features.

Experiments

We design our experiments to answer the following questions:

- How does the feature-based adaptation (CORAL) perform compared to the parameter-based adaptation (self-training)?
- How does the hybrid feature-parameter adaptation approach compare with the individual feature-based and parameter-based approaches?

We perform the following experiments, with self-training approach parameters from subsection 5.2.5 with the fixed number of iterations.

1. Run Variance Threshold (VT) on the combined dataset of D_S and D_T to select a subset of features. Run CORAL with the selected features to transform the source. Use the transformed source to learn a Gaussian Naive Bayes classifier. We refer to this experiment as NB+CORAL+VT.

2. Run Bernoulli Naive Bayes with self-training on the source with the original features. We refer to this experiments as NBST.

3. Run Variance Threshold (VT) on the combined dataset of D_S and D_T to select a subset of features. Run Bernoulli Naive Bayes with self-training on the source represented using the features selected with VT. We refer to this experiments as NBST+VT.

4. Run Variance Threshold (VT) on the combined dataset of D_S and D_T to select a subset of features. Run CORAL with the selected features to transform the source. Run Gaussian Naive Bayes with self-training on the transformed source. We refer to this experiment as NBST+CORAL+VT.

We setup the experiments same as in Subsection 5.2.5 except for feature selection.

- We use the same 11 source-target pairs as in Table 5.2, and split target data into 5-folds, each source is "aligned" with three target unlabeled folds, one target fold is used for testing, and one target fold is kept for future use as potential target labeled data. Similarly, NBST uses three target unlabeled folds in the training process, and one fold for testing.

- The number of features selected varies from one experiment/split to another. For example, for one pair, the original dataset has 1334 features, and the VT approach selects anywhere from 160 to 176 features (for different splits), thus resulting in significant dimensionality reduction.

- In preliminary work, we varied the value of the threshold k in VT. Precisely, we experimented with $k = 0.95, k = 0.90, k = 0.80$, etc. The highest accuracy was obtained when the threshold is equal to 0.99, and this is the value used in the experiments.

- We varied the number of instances in the sources to see how the performance vary with different numbers of source instances. Concretely, in addition to the whole source labeled data, which we denote as Total, we also selected 500, 1000, 2000 instances from each class (on-topic or off-topic), respectively.

The evaluation of the classifiers is based on accuracy as the dataset is relatively balanced.

5.3.3 Results and discussion

The results of the experiments that are used to compare NBST, NBST+VT, and CORAL+VT, NBST+CORAL and NBST+CORAL+VT are shown in Table 5.10. In the table, a source and target pair (S→T) is denoted using the source and target disaster abbreviations introduced in Table 5.2. The numbers 500, 1000, 2000 in the header denote how many source instances from each class (on-topic or off-topic) are used for training, and Total means that all source instances are used. The highlighted values are the best values for each pair with a certain number of source instances across different experiments/approaches. The more highlighted values one approach has, the better that approach performs. We will use this table to answer the research questions.

How does the feature-based adaptation (CORAL) perform compared to the parameter-based adaptation (self-training)?

Based on results in Table 5.10, we compare two domain adaptation methods, self-training and CORAL with Naive Bayes as a base classifier, i.e., NB+CORAL+VT and NBST+VT. As can be seen, NBST+VT performs better than NB+CORAL+VT overall, and implicitly better than the baselines (NB, NB+VT). Furthermore, NBST is better than CORAL, as NBST benefits from using all features, while CORAL works better with a selected set of features.

Table 5.10: Accuracy results of domain adaptation approaches, Naive Bayes with self-training (NBST), and Naive Bayes with self-training and Variance Threshold (VT) feature selection (NBST+VT), CORAL with VT feature selection (NB+CORAL+VT), and the hybrid feature-parameter based approach (self-training on top of CORAL) with VT feature selection (NBST+CORAL+VT)

	$S \rightarrow T$	500	1000	2000	Total		$S \rightarrow T$	500	1000	2000	Total
NB ST	$SH \rightarrow QF$	82.4	84.3	85.5	82.4	NB + $CORAL$ + VT	$SH \rightarrow QF$	75.1	85.1	85.2	83.8
	$SH \rightarrow BB$	**82.8**	**83.2**	**84.6**	**84.1**		$SH \rightarrow BB$	78.9	76.4	82.7	76.6
	$QF \rightarrow BB$	78.7	79.2	81.4	**81.9**		$QF \rightarrow BB$	83.4	81.9	80.4	68.0
	$SH \rightarrow WT$	92.3	92.4	92.3	90.8		$SH \rightarrow WT$	80.2	73.8	67.4	83.6
	$BB \rightarrow WT$	92.9	92.8	93.9	94.8		$BB \rightarrow WT$	88.8	94.5	94.5	94.9
	$SH \rightarrow OT$	85.8	**87.9**	**88.7**	**87.8**		$SH \rightarrow OT$	85.3	85.8	85.7	75.3
	$QF \rightarrow OT$	**85.3**	85.0	85.5	**85.5**		$QF \rightarrow OT$	82.7	86.7	87.3	81.5
	$BB \rightarrow OT$	**84.8**	**84.8**	**86.5**	**86.9**		$BB \rightarrow OT$	79.2	85.3	82.7	82.3
	$SH \rightarrow AF$	79.3	79.9	84.0	82.6		$SH \rightarrow AF$	77.0	84.3	85.7	84.6
	$QF \rightarrow AF$	82.4	83.0	**85.2**	**86.0**		$QF \rightarrow AF$	73.2	80.7	81.3	80.2
	$BB \rightarrow AF$	80.2	**82.4**	**83.9**	**84.0**		$BB \rightarrow AF$	71.2	71.3	74.4	79.1
	Average	84.3	85.0	**86.5**	**86.1**		Average	79.5	82.3	82.5	80.9
NB ST + VT	$SH \rightarrow QF$	**84.4**	84.5	83.9	80.6	$NBST$ + $CORAL$ + VT	$SH \rightarrow QF$	81.5	**89.1**	**88.0**	**87.7**
	$SH \rightarrow BB$	82.7	83.0	84.5	81.5		$SH \rightarrow BB$	75.6	76.5	82.4	76.3
	$QF \rightarrow BB$	79.8	80.5	81.2	**81.9**		$QF \rightarrow BB$	**84.5**	**85.9**	**84.8**	74.3
	$SH \rightarrow WT$	**93.6**	**93.6**	**92.8**	**92.3**		$SH \rightarrow WT$	80.5	88.1	79.1	90.6
	$BB \rightarrow WT$	**94.4**	**94.6**	**95.0**	**95.3**		$BB \rightarrow WT$	87.5	92.1	94.5	93.6
	$SH \rightarrow OT$	**86.8**	87.4	87.5	86.6		$SH \rightarrow OT$	86.3	87.8	88.0	73.5
	$QF \rightarrow OT$	83.2	83.5	84.3	84.2		$QF \rightarrow OT$	82.1	**88.7**	**87.8**	81.5
	$BB \rightarrow OT$	84.5	84.0	84.4	84.3		$BB \rightarrow OT$	82.8	82.3	85.4	82.1
	$SH \rightarrow AF$	81.5	81.8	81.8	79.7		$SH \rightarrow AF$	**84.9**	**87.6**	**87.9**	**86.1**
	$QF \rightarrow AF$	**82.7**	**83.2**	84.6	85.1		$QF \rightarrow AF$	75.1	82.4	83.0	83.5
	$BB \rightarrow AF$	**80.5**	80.5	81.7	82.2		$BB \rightarrow AF$	74.7	79.7	75.7	81.1
	Average	**84.9**	85.1	85.6	84.9		Average	81.4	**85.5**	85.1	82.8

How does the hybrid feature-parameter adaptation approach compare with the individual feature-based and parameter-based approaches?

To answer this question, we compare the results of NBST+CORAL+VT with the results of NBST and NBST+VT, and also with the results of NB+CORAL+VT in Table 4. Overall the self-training approach performs better than the hybrid feature-parameter variant NBST+CORAL+VT. However, for some specific source-target pairs, the combined self-training and CORAL approach is better than either self-training or CORAL alone. More concretely, we can see that for pairs SH→QF, QF→BB, SH→AF, the hybrid approach achieves the best performance. Moreover, NBST+CORAL+VT can improve the performance of NB+CORAL+VT for almost all pairs, regardless of the number of source

instances used. As last, NBST+CORAL+VT performs well also in terms of precision and recall metrics too, with an average precision of 0.836 and average recall of 0.827 when all source instances are used for training, which we didn't show here.

5.4 Conclusions

In this chapter, we proposed domain adaptation approaches based on Naive Bayes with Expectation Maximization (EM) and self-training strategy, we did hyper-parameter analysis to provide guidance for potential practical application. We further compared the self-training approach with a feature representation based approach, Correlation Alignment (CORAL), on top of supervised Naive Bayes classifiers, and proposed a hybrid model to combine self-training and CORAL on top of Naive Bayes models. We experimented with different source and target disaster pairs to identify disaster related tweets.

EM and self-training incorporate labeled data from a source disaster and unlabeled data from an emerging target disaster into a classifier for the target disaster. We first compared Naive Bayes with self-training (NB-ST) with supervised classifiers learned only from source (NB-S) and with domain adaption classifiers based on expectation-maximization (NB-EM) in Section 5.2. The results of our experiments showed that using source data only with supervised learning could help when the source and target disasters were similar. However, the domain adaptation approaches were always better than the supervised learning with source data only. Between the NB-ST and NB-EM approaches, generally the NB-ST approach was better. As last, our experimental results showed that the domain adaptation approaches could give results comparable, and in some cases better, than an ideal supervised classifier that would have (noisy) labeled target data available. However, in general, there is still room for improving the results of domain adaptation classifiers as compared to the ideal supervised classifier that would have access to labeled target data.

In subsection 5.2.5, we provided recommendations for good overall parameter values to be used in practice for the domain adaptation algorithm with self-training. Furthermore, our study provided recommendations with respect to the number of labeled source instances to

be used for good performance, and also with respect to the number of iterations needed for good performance. We showed that in general more source labeled data could produce better results, but performance did not change significantly after a certain amount, for example after 4000 instances (2000 on-topic and 2000 off-topic). If that amount of source labeled data is not available, our experiments showed that even 500 source labeled instances could result in a relatively good classifier. This provides some insights into how much effort we should put into labeling data. Based on the analysis of performance variation with parameters, we recommend small values for δ (e.g., 0.001) to very slowly shift the weight from source to target and get good overall results. Furthermore, our study suggested that adding more than 1 instances from each class at each iteration of the algorithm (e.g., adding 5 or 10 instances from each class) benefited the final classifier. Finally, we showed that best overall results were obtained when fixing the shifting speed δ as 0.001, adding $k = 5$ instances from each class at each iteration, and running 300 iterations. This suggests that we can potentially use these parameter values as default in practice without sacrificing performance, and thus help disaster management and response teams prioritize the information that they need to more carefully analyze.

In Section 5.3, we compared a feature-based adaptation approach, CORAL, with self-training and also combined these two in one model on the same dataset as in the previous sections. CORAL is a simple yet effective domain adaptation method based on unsupervised feature alignment between source and target data. Experimental results showed that Naive Bayes with self-training (NBST) performed better than Naive Bayes with CORAL in many cases, especially when leveraging more features. It could be hypothesized that the gradual labeling and usage of the target data in NBST could potentially capture more knowledge from the target unlabeled data, as compared to the one-shot use of the target unlabeled data in CORAL (to shift the distribution of the source data). The comparison of the hybrid approach with the individual feature-based and parameter-based adaptation approaches supported this hypothesis. Specifically, when comparing the hybrid approach with the individual feature-based and parameter-based approaches, no significant gains were observed from the combination. Furthermore, the results of the hybrid were overall similar with the

results of the NBST approach itself. This suggests that the CORAL approach, while useful on its own, did not have complementary strengths as compared with NBST for this dataset. Nevertheless, there were some specific pairs where the hybrid approach performed better than the self-training approach.

There are of course limitations to our work. To understand how the algorithm behaviors generalize to different classification tasks, the approaches in this chapter for example self-training domain adaptation approach needs to be analysed with more classification tasks and also multi-class classification tasks. For example, we can consider the task of classifying disaster related tweets further into subcategories based on the user publishing the tweets or based on the tweet content - a tweet can be contributed by eyewitnesses, by victims or by news agents, or a tweet can be about casualty, infrastructure damage or volunteer and donation needs.

Chapter 6

Domain adaptation using embedding-based approaches

In this chapter, we investigate embedding-based approaches that use pre-existing embedding models to transfer knowledge. The general goal is to build classifiers that are ready-to-use whenever a disaster strikes. Concretely, we experiment with embeddings on word-level, sentence-level, and also pre-trained language model based contextual embeddings under the multi-source domain adaptation setting. These approaches can be viewed as feature/representation based domain adaptation approaches. In Section 6.1, we experiment with simple adaptation approaches with pre-trained word embeddings and sentence encoding models. We evaluate how different representations using pre-trained word embeddings or sentence encodings perform with supervised learning models on the three datasets introduced in Chapter 4. In Section 6.2, we evaluate pre-trained word embedding with deep learning models. We also propose to run self-training on top of a CNN model. In Section 6.3, we experiment with approaches based on the state-of-the-art pre-trained language models, specifically BERT, and apply a multi-source domain adversarial model using the representations computed from pre-trained BERT model.

6.1 Pre-trained word/sentence embeddings with traditional supervised models

6.1.1 Representations using word embeddings

Currently, research on word embeddings is still one of the most popular topics in the NLP area. Here, we adopt three types of word embeddings widely used in the NLP community:

1. *Word2Vec* (Mikolov et al., 2013a,b);
2. *GloVe* (Pennington et al., 2014);
3. *FastText* (Bojanowski et al., 2017; Mikolov et al., 2018).

With each type, we use both existing embeddings pre-trained on Wikipedia or Twitter data, and also crisis word embeddings trained specifically on our crisis tweet corpora. Subsequently, we use several simple ways to combine the word embeddings into reduced representations, as described below:

a) *Mean*: We average the embeddings of each word in the tweet along each dimension. Thus, a tweet vector will have the same dimension as a word vector/embedding.

b) *MinMaxMean* (MMM): In addition to mean, we also take the minimum and maximum over all the words in a tweet, along each dimension of the word vectors. Each aggregation, min/max/mean, will produce a vector that has the same dimension as the word vectors. We concatenate the vectors corresponding to min/max/average, respectively, and obtain a tweet vector whose dimension is three times the dimension of the word vector.

c) *Tf-idf-Mean*: We assign Tf-idf (term frequency - inverse document frequency) weights to the words in a tweet, and calculate the weighted average of the word embeddings along each dimension (where the contribution of a word is proportional to its Tf-idf weight).

6.1.2 Representations using sentence encodings

Given that tweets resemble sentences as they are relatively short, we also experiment with approaches on sentence-level embeddings/encodings. Specifically, we use the following models to get tweet representations:

1. *Smooth Inverse Frequency* (SIF) (Arora et al., 2017): The representation produced by this approach can be seen as the weighted average of the word vectors, modified by removing the projections of the average vectors on their first principal component ("common component removal') (Arora et al., 2017). This simple method has been shown to beat more sophisticated models, such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), on semantic textual similarity tasks.

2. *InferSent* (Conneau et al., 2017): This approach consists of universal sentence representation models trained with natural language inference data, using different network architectures such as Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRU), or Bi-directional LSTM networks (BiLSTM). We use the pre-trained model published by the authors to generate tweet representations with this approach.

3. *Universal Sentence Encoder on TensorFlow* (tfSentEncoder) (Cer et al., 2018): The universal sentence encoding models are trained with the same data as the InferSent models, but with different model architectures.

In general, the purpose and usage of the universal sentence encoders are similar to purpose/usage of the word embeddings, as the pre-train universal sentence encoders can help with NLP and text classification tasks that rely on sentences (Cer et al., 2018).

6.1.3 Datasets and preprocessing

We use all three datasets introduced in Chapter 4 here, specifically: 1) CrisisLexT6 (Olteanu et al., 2014); 2) CrisisLexT26 (Olteanu et al., 2015); and 3) 2CTweets (Schulz et al., 2017).

To benefit the most from pre-trained embeddings, ideally, one should preprocess the data to be embedded the same way as the corpus that was used for training the embeddings.

Table 6.1: Statistics about the datasets (CrisisLexT6, CrisisLexT26, and 2CTweets), before and after cleaning

	Before Cleaning			After Cleaning		
CrisisLexT6	On-topic	Off-topic	Total	On-topic	Off-topic	Total
2012_Sandy_Hurricane	6138	3870	10008	5443	3757	9200
2013_Queensland_Floods	5414	4619	10033	3324	4530	7854
2013_Boston_Bombings	5648	4364	10012	4824	4301	9125
2013_West_Texas_Explosion	5246	4760	10006	4123	4711	8843
2013_Oklahoma_Tornado	4827	5165	9992	4101	5111	9212
2013_Alberta_Floods	5189	4842	10031	4550	4745	9295
CrisisLexT26	Informative	Non-Inf.	Total	Informative	Non-Inf.	Total
2012_Colorado_wildfires	685	268	953	665	252	917
2013_Queensland_floods	728	191	919	681	183	864
2013_Boston_bombings	417	512	929	397	489	886
2013_West_Texas_explosion	472	439	911	444	390	834
2013_Alberta_floods	685	298	983	665	284	949
2013_Colorado_floods	768	157	925	736	147	883
2013_NY_train_crash	904	95	999	684	88	772
2CTweets	Yes	No	Total	Yes	No	Total
Memphis	361	721	1082	333	699	1032
Seattle	800	1404	2204	739	1293	2032
NYC	413	1446	1859	373	1411	1784
Chicago	214	1270	1484	202	1254	1456
San Francisco	304	1176	1480	290	1146	1436
Boston	604	2216	2820	586	2123	2709
Brisbane	689	1898	2587	667	1746	2413
Dublin	199	2616	2815	189	2574	2763
London	552	2444	2996	490	2287	2777
Sydney	852	1991	2843	832	1947	2779

However, for the pre-trained Word2Vec and FastText embeddings, the original preprocessing performed is not well documented, so we use directly the raw tweets to obtain the corresponding word embeddings representations. As opposed to that, the preprocessing script used when training GloVe word embeddings on Twitter data is available from the GloVe's website[1]. Thus, we apply the same preprocessing for all our datasets when using pre-trained GloVE or crisis-specific word embeddings. Specifically, we use a Python version of the GloVe's Ruby preprocessing script, which consists of the following main steps: 1) replacing URLs and user mentions with placeholders <url> and <user>, respectively; 2) replacing different emoticons with placeholders such as <smile>, <lolface>, <sadface>, <neutralface>

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

and <heart>, respectively; 3) replacing numbers with a placehoder <number>; 4) changing uppercase words to lowercase words, and tagging them with the tag <allcaps>, for example, "HURRICANE" becomes "hurricane <allcaps>"; 5) similarly, tagging punctuation repetitions, elongated words and hashtags, for example "!!!" becomes "! <repeat>", "soooo" becomes "so <elong>" and "#HurricaneSandy" becomes "<hashtags> hurricanesandy"; 6) converting all tweets to lowercase and tokenizing them based on whitespace using the Stanford Tokenizer. Finally, duplicate tweets identified after preprocessing are removed. The statistics of each class in each event dataset, before and after the preprocessing, together with the total number of tweets in the dataset, are shown in Tables 6.1 for the three datasets, respectively.

Our crisis tweet corpora for training crisis specific word embeddings are also processed the same way. Besides the three datasets described above, the corpora also includes tweets that we collected through the Twitter Streaming API during several disasters that happened in 2017 hurricane season, specifically Hurricane Harvey, Hurricane Irma, Hurricane Maria, and also Mexico Earthquake. The total corpora contains approximately 5.8 million tweets. Most of these tweets are crisis related, but there is also some inherent noise due to the fact that the streaming was keyword-based.

6.1.4 Experimental setup

Once we obtain the vector representations for tweets, any supervised learning algorithm (or even more complex domain adaptation algorithms) can be used to learn classification models using the tweet reduced representations. Here, for the sake of simplicity, and also to satisfy the real-time prediction requirements, we choose first to experiment with four traditional supervised machine learning algorithms:

1. Naive Bayes (NB);
2. Random Forest (RF);
3. K-Nearest Neighbors (KNN);

4. Support Vector Machines (SVM).

We included the Naive Bayes algorithm in our study as it doesn't require hyper-parameter tuning, and Random Forest, KNN, and especially SVM, all have been extensively used in text classification tasks.

For each of the three datasets, we experimented with three types of word embeddings, both existing pre-trained embeddings, and custom embeddings trained on crisis tweets, as well as three sentence embedding models, for a total of nine different representations for the tweets. We include three traditional bag-of-words representations in the experiments to serve as baselines, 0/1 binary representation, words counts and tf-idf representations. Notice that Multivariate Bernoulli Naive Bayes model is used for 0/1 binary bag-of-word representation, Multinomial Naive Bayes is used for count and tf-idf representations and Gaussian Naive Bayes is used for word embeddings representations. We take target data as unlabeled data and use words from it as in bag-of-words experiments, specifically, we use 5000 unigrams to represent a tweet in these baselines[2]. In word embeddings experiments, all words from sources and target disasters are used. Embeddings of words that are not included in pre-trained word embeddings vocabulary are set to zero vectors. and use all the words of that tweet that have pre-trained embeddings.

The notations and details of the embeddings used are as follows:

- *Word2Vec*: denotes the pre-trained Word2Vec embeddings. Specifically, we used the set of embeddings trained on Google news. The dimension of the Word2Vec embeddings is 300.

- *CrisisW2V*: denotes the Word2Vec embeddings trained with our crisis tweet corpus. We use the Gensim package implementation of Word2Vec, and trained a CBOW (Continuous Bag Of Words) model, also with dimension 300. We use the default values for all the other parameters in Gensim package.

[2]We also experiment with using all words and also unigrams together with bigrams, the results are worse than 5000 unigrams in general.

- *GloVe*: denotes the GloVe embeddings pre-trained on Twitter data. We also experimented with GloVe embeddings pre-trained on Wikipedia data, but the results were worse than those obtained with the Twitter embeddings, we will not show these here. Furthermore, there are four different sets of pre-trained GloVe Twitter embeddings, corresponding to four dimensions: 25, 50, 100 and 200. Our preliminary results showed that 50 or 100 dimensions give similar results, generally better or comparable with the results obtained with 25 or 200 dimensions. We only show the results obtained with embeddings with 100 dimensions.

- *CrisisGloVe*: denotes GloVe embeddings trained on our crisis tweet corpus. We use the GloVe original package to learn the embeddings, minimum word frequency count is 10, maximum iterations is set to 100 and window size is 10. We only show the results obtained with the 100 dimensional embeddings with same reason as *GloVe* setting.

- *FastText*: denotes the pre-trained FastText embeddings. We experiment with the 1 million word vectors trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset. The dimension is 300.

- *CrisisFastText*: denotes the embeddings trained on the crisis tweet corpus with the FastText original package. We used CBOW as in CrisisW2V, with dimension 300, and default values for all the other parameters.

- *SIF*: denotes the SIF approach, which is considered to be a baseline for sentence embeddings. The original paper used GloVe embeddings pre-trained on the Common Crawl data. We will use GloVe embeddings with 100 dimensions, pre-trained on Twitter data here, for consistency with the *GloVe* setting.

- *InferSent*: denotes the universal sentence representation model trained with natural language inference data. We used the pre-trained model (Conneau et al., 2017), which encodes tweets into 4096 dimensional vectors.

- *tfSent*: denotes the universal sentence encoder from (Cer et al., 2018). We use the

encoder available from TensorFlow TF-hub, which encodes tweets into 512 dimensional vectors.

As discussed in Section 6.1.3 (Dataset section), all experiments are performed on processed tweets, except for the experiments with *Word2Vec* and *FastText* which are performed on original raw tweets corresponding to the cleaned tweets, due to the fact that the preprocessing steps are not explicit for these two types of pre-trained embeddings[3]. For each type of word embedding, we further used the three aggregation approaches mentioned in the previous section, specifically, Mean, Min/Max/Mean (MMM) and Tf-idf-Mean (Tf-idf) to convert the word embeddings into reduced tweet representations. Furthermore, for each word-based or sentence-based representation, we experimented with four supervised learning algorithms from the scikit-learn library: 1) Gaussian Naive Bayes (GNB); 2) Random Forest (RF), where the number of trees (n_estimators) was set to 100 (and default values were used for the other parameters); 3) K Nearest Neighbors (KNN), for which the default number of neighbors was 5; 4) Support Vector Machine (SVM), with default parameters, including cost parameter $C = 1$ and RBF kernel.

Leave-one-out: For CrisisLexT6 and 2CTweets, the classification task is to separate disaster/incident relevant tweets from the irrelevant ones. After that, we can further filter out the informative ones from the relevant tweets. This classification task is evaluated with CrisisLexT26. We could combine, for example, CrisisLexT6 and 2CTweets in the future. We evaluate different tweet representations based on word embeddings or sentence embeddings, on each of the three datasets. A leave-one-out setting is used for evaluation to simulate a real scenario. Namely, for each dataset, in a particular experiment, we select one event as the target, and use the rest of the events from that dataset as training. For example, when Hurricane Sandy from CrisisLexT6 is selected as test, the other five disasters from CrisisLexT6 are used for training. Each disaster is left out in one experiment, therefore, the number of experiments conducted for one dataset is given by the number of events in the dataset. The results reported for a dataset are averaged over all experiments conducted on

[3]We also run these experiments on the cleaned tweets and the results were slight better or worse depending on the datasets, and therefore not showed here.

Table 6.2: Average accuracy values (and standard deviation) of CrisisLexT6 using bag-of-words representations (baseline) and word embeddings representation with traditional models. For word embeddings representations, the best value of each column is highlighted in bold font. Furthermore, underscored values represent the best values among those corresponding to a particular classifier. The more underscored values in a column, the better the corresponding word embedding performs as compared to other embeddings. Also, the more bold values one type classifier has for a dataset, the better that type of classifier is for that dataset.

CrisisLexT6		Binary		Count		Tf-idf	
NB		85.8±9.4		84.3±9.0		84.5±9.7	
RF		81.3±10.8		81.4±10.3		81.4±10.1	
KNN		69.4±7.9		71.6±7.6		78.5±9.6	
SVM		64.2±15.1		64.6±15.0		74.7±9.4	
		Word2Vec	CrisisW2V	GloVe	CrisisGloVe	FastText	CrisisFastText
GNB	Mean	80.8±7.9	75.7±7.3	83.4±7.6	78.3±8.5	82.5±7.2	76.4±5.9
	MMM	79.7±6.2	79.8±5.8	82.8±6.7	79.1±6.5	79.8±6.0	77.4±6.0
	Tf-idf	68.7±4.7	75.3±8.8	83.1±5.9	78.5±10.3	66.2±4.6	75.2±6.5
RF	Mean	84.1±8.7	83.3±9.2	87.5±7.4	**83.5±10.7**	84.7±9.3	81.5±8.5
	MMM	82.4±9.1	83.2±8.8	85.9±9.6	81.6±11.6	80.8±9.5	81.9±7.4
	Tf-idf	82.4±8.7	82.1±10.1	87.2±7.1	82.7±12.1	82.4±9.4	80.5±9.7
KNN	Mean	84.7±4.1	82.0±6.0	83.8±3.4	82.5±8.2	82.3±3.5	77.9±4.8
	MMM	86.3±5.8	82.1±6.7	84.8±5.9	82.9±9.9	85.0±5.9	74.6±4.3
	Tf-idf	80.1±3.7	80.1±5.1	81.9±3.5	80.0±6.9	79.1±3.4	75.1±4.4
SVM	Mean	84.6±7.7	**86.3±10.2**	**89.4±4.7**	82.6±12.5	82.9±8.1	**84.7±11.0**
	MMM	**87.3±6.6**	85.8±9.5	89.0±5.9	83.3±13.1	86.2±7.8	83.7±8.3
	Tf-idf	87.2±5.7	79.5±11.2	88.8±4.3	82.8±12.4	**87.5±5.4**	79.2±10.4

that dataset. Intuitively, the averages should provide an indication of how well the models built generalize to different future events.

6.1.5 Results and discussion

Given the variety of word embeddings that are available in the machine learning and NLP community, we aim to understand how the performance of traditional classifiers vary with different types of word embeddings, and whether re-training the embeddings on crisis data is necessary or not. Furthermore, given that text classification tasks (e.g., sentiment analysis)

Table 6.3: Average accuracy values (and standard deviation) of CrisisLexT26, using bag-of-words representations(baseline) and word embeddings representation with traditional models. For word embeddings representations, the best value of each column is highlighted in bold font. Furthermore, underscored values represent the best values among those corresponding to a particular classifier. The more underscored values in a column, the better the corresponding word embedding performs as compared to other embeddings. Also, the more bold values one type classifier has for a dataset, the better that type of classifier is for that dataset.

CrisisLexT26		Binary		Count		Tf-idf	
NB		78.5±6.5		78.4±6.3		83.0±4.8	
RF		84.2±5.0		84.7±5.0		84.0±5.5	
KNN		45.0±10.2		54.3±8.7		79.9±5.9	
SVM		84.2±5.9		84.0±6.0		70.2±14.8	
		Word2Vec	CrisisW2V	GloVe	CrisisGloVe	FastText	CrisisFastText
NB	Mean	80.7±5.8	79.6±6.0	82.1±6.1	81.7±6.6	79.8±6.2	78.5±5.8
	MMM	78.7±6.7	79.6±6.2	84.1±4.6	83.2±4.8	79.3±6.0	77.3±5.1
	Tf-idf	80.6±5.6	81.4±5.3	81.2±5.3	81.1±6.6	80.8±5.7	81.5±5.0
RF	Mean	83.6±4.5	84.9±4.6	85.0±4.6	85.8±4.0	84.3±4.7	84.1±4.8
	MMM	83.8±4.6	85.3±4.3	85.6±4.5	86.1±4.6	**84.4±5.0**	84.3±4.2
	Tf-idf	82.4±4.6	85.0±4.2	83.0±4.6	84.5±4.7	81.6±4.6	84.2±4.0
KNN	Mean	82.3±6.0	83.0±5.0	83.6±5.6	83.4±5.5	81.3±7.0	82.6±4.4
	MMM	80.5±6.3	80.0±3.9	83.8±5.8	82.2±5.7	80.9±6.5	76.9±6.2
	Tf-idf	79.7±6.1	83.0±4.8	83.0±5.2	82.1±6.7	79.6±7.4	84.5±4.4
SVM	Mean	70.3±14.8	**86.2±3.9**	85.7±4.5	**86.3±4.2**	70.2±14.8	**85.5±4.2**
	MMM	79.7±8.2	85.7±3.8	**86.1±4.6**	86.1±4.1	71.2±14.3	84.1±4.8
	Tf-idf	**84.4±4.6**	83.8±4.5	84.1±6.3	84.1±6.0	83.9±4.3	83.8±4.8

Table 6.4: Average accuracy values (and standard deviation) of 2CTweets, using bag-of-words representations(baseline) and word embeddings representation with traditional models. For word embeddings representations, the best value of each column is highlighted in bold font. Furthermore, underscored values represent the best values among those corresponding to a particular classifier. The more underscored values in a column, the better the corresponding word embedding performs as compared to other embeddings. Also, the more bold values one type classifier has for a dataset, the better that type of classifier is for that dataset.

2CTweets		Binary		Count		Tf-idf	
NB		89.4±4.5		88.8±4.5		89.4±4.1	
RF		89.9±3.4		90.0±3.4		89.3±3.8	
KNN		82.1±7.3		81.7±7.3		86.6±5.2	
SVM		90.9±3.3		90.4±3.6		77.3±8.5	
		Word2Vec	CrisisW2V	GloVe	CrisisGloVe	FastText	CrisisFastText
NB	Mean	88.1±4.6	78.5±5.8	85.0±5.3	82.9±5.3	84.9±4.7	73.7±5.6
	MMM	87.7±5.1	80.1±5.7	84.4±5.6	82.8±6.1	83.8±5.3	74.4±5.9
	Tf-idf	82.5±5.4	81.8±4.8	85.2±5.3	85.4±5.0	78.6±5.5	79.1±5.4
RF	Mean	89.6±4.2	88.6±3.9	89.0±3.7	89.1±3.3	88.6±4.3	87.7±4.5
	MMM	**90.5±3.6**	90.5±3.6	**90.6±3.3**	90.3±3.2	**90.5±3.8**	88.3±4.5
	Tf-idf	86.4±5.6	88.5±4.0	88.4±4.3	87.6±4.6	85.7±5.9	87.6±4.4
KNN	Mean	90.1±3.2	89.1±3.6	88.0±3.9	88.7±4.3	87.9±4.1	86.9±4.6
	MMM	90.0±3.3	88.2±4.1	87.1±4.9	88.5±4.2	88.1±2.8	82.2±4.7
	Tf-idf	88.4±3.2	88.1±4.1	87.5±4.0	87.6±4.4	87.0±4.4	86.6±4.4
SVM	Mean	85.9±4.5	90.7±3.6	89.3±3.6	89.8±3.4	77.7±8.2	**91.2±3.1**
	MMM	89.0±3.7	**92.1±2.5**	90.4±3.7	90.2±3.6	84.0±5.2	89.2±4.0
	Tf-idf	90.3±3.8	88.9±4.7	90.0±3.9	**90.6±3.4**	89.2±4.1	87.6±6.0

Table 6.5: Average accuracy values (and standard deviation) for three datasets with three types of sentence embeddings and traditonal models. For each dataset, the best value of each column is highlighted in bold font. Furthermore, underscored values represent the best values among those corresponding to a particular classifier. The more underscored values in a column, the better the corresponding word embedding performs as compared to other embeddings. Also, the more bold values one type classifier has for a dataset, the better that type of classifier is for that dataset.

		SIF	InferSent	tfSent
CrisisLexT6	NB	83.4±6.5	80.6±6.0	88.1±6.1
	RF	88.5±5.7	**88.3±4.8**	87.9±6.5
	KNN	85.6±3.5	87.3±4.8	87.7±5.2
	SVM	**89.6±4.2**	87.0±6.1	**89.3±5.0**
		SIF	InferSent	tfSent
CrisisLexT26	NB	80.0±2.0	82.1±4.9	80.0±5.3
	RF	**81.3±7.0**	**85.3±4.4**	**84.7±4.1**
	KNN	81.0±6.0	82.2±5.2	81.8±7.1
	SVM	81.0±8.5	70.2±14.8	70.2±14.8
		SIF	InferSent	tfSent
2CTweets	NB	83.5±3.2	85.9±5.0	88.0±5.0
	RF	**88.1±5.3**	89.8±3.9	**89.5±4.6**
	KNN	87.6±3.0	**90.1±3.6**	88.7±3.6
	SVM	88.0±4.6	84.9±5.5	86.4±7.0

have been shown to benefit from sentence-level embeddings pre-trained on data from other NLP tasks, we also aim to understand how different sentence-level embeddings perform when used for crisis tweet classification tasks. Finally, towards the ultimate goal of building "generalized" classifiers, we aim to gain insights into how different classifiers handle different types of embeddings, and what algorithms best handle the embeddings, in general. The observations that we seek to draw from our experiments could be used to provide guidelines for future studies that need to choose among different types of embeddings and supervised classifiers, as well as guidelines for the adoption of the framework for domain adaptation through embeddings in practice.

The results of the experiments that used tweet representations based on word embeddings are shown in Table 6.2, 6.3 and 6.4, and the results based on sentence embeddings are shown in Table 6.5. First of all, for all three datasets, we can always find a model trained word embeddings representations outperform the bag-of-words representations baselines. Although, the baselines of 2CTweets dataset are very strong, averaged accuracy of SVM model up to 90%, but the averaged accuracy of CrisisWord2Vec MinMaxMean representation with SVM reaches 92%. We will focus on analyzing the results of models trained with pre-trained word embeddings in the following. Specifically, the analysis is driven by several research questions described below.

Among the pre-trained word embeddings (i.e., Word2Vec, GloVe and FastText), which embeddings and aggregations work better, in general, with different supervised classifiers?

By analyzing the results in Table 6.2, 6.3 and 6.4 by column, we can compare the performance of different types of embeddings and aggregation methods (i.e., Mean, MinMaxMean or Tfidf weighted averaging), when used with different algorithms. For each dataset and each classifier, we underscore the best value obtained with that classifier on the dataset to observe which types of embeddings/aggregations perform well for specific classifiers and across different classifiers. The more underscored values in a column, the better the corresponding embedding performs. From Table 6.2, 6.3 and 6.4, we can see that the GloVe embeddings work better for two datasets out of three. Specifically, for CrisisLexT6, GloVe pre-trained Twitter embeddings work the best, while for CrisisLexT26 dataset, CrisisGloVe embeddings

are better. However, for the 2CTweets dataset, the Word2Vec embeddings (either pre-trained or trained on crisis data) are better across several supervised classifiers, although the performance of the GloVe embeddings is very close or even better in some cases (e.g., with Random Forests). One possible explanation for this may be that the 2CTweets dataset is mostly about incidents that appear as local news in a city, as opposed to large-scale disasters. Intuitively, the GloVe embeddings pre-trained on Twitter may capture well disasters such as Hurricane Sandy (given that tweets about disasters may be part of the training set), but may not capture well the local incidents.

Regarding the different approaches to aggregate word embeddings, the performances vary with the aggregation approach, and it cannot be claimed that one approach is better than the others, in general. However, if we focus on the best types of word embeddings for each dataset (CrisisLexT6, GloVe column; CrisisLexT26, CrisisGloVe column; and 2CTweets, Word2Vec, CrisisW2V columns), it can be observed that the MinMaxMean (MMM) is generally better than or very close to the other two aggregation approaches, although sometimes the best values are achieved by Mean aggregation.

Given the existing pre-trained embeddings, do crisis tweet classification tasks benefit from embeddings trained specifically on a crisis tweet corpus?

When comparing the pre-trained embeddings with the embeddings trained on the crisis tweet corpus, we can see that for CrisisLexT6 the embeddings pre-trained on a general corpus are better, while for CrisisLexT26 and 2CTweets the embeddings trained on the crisis tweet corpus are better. In particular, for CrisisLexT26, the results with CrisisW2V and CrisisFastText are better for almost all classifiers, as compared to the results obtained with the corresponding Word2Vec and FastText pre-trained embeddings. While the CrisisGloVe embeddings are not always better than the pre-trained Twitter GloVe embeddings, they can still achieve competitive performance when used with Random Forest or SVM classifiers.

One possible reason that the GloVe crisis-specific embeddings perform better than the pre-trained embeddings on CrisisLexT26, but not on the CrisisLexT6 dataset, may be the fact that the CrisisLexT6 classification task (relevant to disasters or not relevant) is more general, and thus its vocabulary may be better covered by the general Twitter corpus used

for the pre-trained embeddings. As opposed to that, the CrisisLexT26 tasks are more specific to crises and benefit more from crisis embeddings.

Among the sentence encodings (i.e., SIF, InferSent, tfSent), which encodings work better, in general, with different classifiers? How do the sentence encodings compare with the word embeddings?

By comparing columns in Table 6.5, we can see that the InferSent sentence encoder is better for CrisisLexT26 and 2CTweets, while the simple SIF encoder is generally better for the CrisisLexT6 dataset. InferSent generates sentence encodings with 4096 dimensions, a significantly larger number of dimensions as compared to the SIF sentence encoder (which uses just 100 dimensions), and the tfSent encoder (which uses 512 dimensions). Based on our prior experimentation with CrisisLexT6 and CrisisLexT26, a relatively small number of features is needed to discriminate between relevant and irrelevant tweets in CrisisLexT6, while a larger number of features is needed to discriminate between crisis informative and non-informative tweets in CrisisLexT26. Our prior observations match with the current study which suggests that a sentence encoder that produces vectors with a small number of dimensions (SIF) is useful for CrisisLexT6, while an encoder that produces vectors with a large number of dimensions (InferSent) is useful for CrisisLexT26. Thus, in general, the choice of the sentence encoder may be related to the choice of the number of dimensions used by the encoder, which in turn depends on difficulty of the classification task at hand.

To evaluate word embeddings versus sentence encodings, we compare Tables 6.2, 6.3, 6.4 with 6.5, and observe that the best values for a dataset are generally obtained using word embeddings. This is counter-intuitive, as one would expect the sentence-level encodings to better capture the content of a tweet. The reason for this may be related to the fact that we do not perform hyper-parameter tuning for classifiers such as RF and SVM. For CrisisLexT6, the Naive Bayes (NB) classifier, which does not have any hyper-parameters, the tfSent model produces an average accuracy of 88%, which is a 5% improvement of the best accuracy obtained with word embeddings. Thus, sentence encoders have great potential for crisis tweet classification tasks, but more experiments and hyper-parameter tuning are needed to better evaluate their benefits.

Among the classifiers studied, NB, RF, KNN and SVM, which one benefits more from embeddings?

When using word embeddings, the SVM classifier performs the best overall, regardless of the type of embedding and aggregation employed. The results of the RF classifier are sometimes the best for a dataset, or very close to the results of SVM. When using sentence encodings, both RF and SVM classifiers work well for the CrisisLexT6 dataset, while for the other two datasets, the RF classifier works better. However, we believe that hyper-parameter tuning might improve the results of SVM, making this classifier more competitive also when using sentence encodings. Gaussian Naive Bayes doesn't work well with simple word embedding aggregations, but performs well with sentence encodings on the larger CrisisLexT6 dataset. The results of the KNN classifier are better than the results of Gaussian Naive Bayes, but worse than those of RF and SVM, in general. The less-competitive performance of KNN may be due to the differences between the events in a dataset. Given that we are using several disasters to train a model, and then we test the model on a left-out test disaster, the different disasters used in training may bring in noise with respect to the test disaster. Even though the word embeddings or sentence encodings are meant to bridge the semantic gap between tweets, KNN is still sensitive to noise as it is making its classifications based on the nearest neighbors selected. If the nearest neighbors are noisy, the classification can be wrong. Thus, overall, our study suggests that SVM or RF are good choices as traditional supervised learning classifiers, but hyper-parameter tuning may still be needed to achieve best performance.

6.2 Pre-trained word embeddings with deep learning models and self-training

In this section, we evaluate pre-trained word embeddings with neural network models:

1. A standard neural network with three layers (NN);
2. A Convolutional Neural Network (CNN) based on (Kim, 2014);

3. A one layer bidirectional Long Short-Term Memory (LSTM)Network.

We have introduced these models in Chapter 2. In addition, we also experiment with self-training on top of the CNN model (CNN-ST), as the CNN model has the best overall performance among the above models.

6.2.1 Experimental setup

We will use the same setting as in the previous Section 6.1 and experiment with the six types of word embeddings. We combine all sources into one training set, the simplest combination strategy, and use 80% of all source instances for training and 20% to validate the models in this section. For the neural network model, we use the *Mean* to aggregate the word embeddings. The dimension of the first layer is the same as the dimension as the word embeddings, while the following two layers have 64 and 32 units, respectively. For CNN and LSTM, the pre-trained word embeddings are loaded as the embedding layer. We use the CNN architecture proposed by Kim (2014). We use filter window sizes of 3, 4, 5 with 100 feature maps each, and dropout rate of 0.5. The LSTM model has a 64 dimension bidirectional LSTM layer followed by a 64 units fully connected layer and an output layer. For all three models, ReLu is used as activation function for hidden layers, and sigmoid for final output layer. The mini-batch size is 128, and the optimization is done with the Adam optimizer.

For CNN with self-training, we use soft-labels. The final outputs of the CNN model are interpreted as probabilities. We first train the CNN model with only the sources disasters as training set. Then at each iteration, we use the CNN model from last step to label target unlabeled instance. We then add all target unlabeled instance as positive instances to the training set with sample weights that are based on the probabilities that they are positive. Similarly, we add all target unlabeled instances as negative instances with sample weights to the training set. Finally, we can train a new CNN model. We can iterate these steps for a certain number of iterations. Here, we only run for three iterations[4]. We use a setting

[4]Considering that CNNs are slow and expensive to train compared with Naive Bayes for example, we only

Table 6.6: Average accuracy values (and standard deviation) for three datasets, using the leave-one-out evaluation strategy, with six types of word embeddings and deep learning models. The best values of each model (row) is highlighted in bold font and the overall best value for the dataset is in bold and underscore.

CrisisLexT6	Word2Vec	CrisisW2V	GloVe	CrisisGloVe	FastText	CrisisFastText
SVM-Mean	84.6±7.7	86.3±10.2	**89.4±4.7**	82.6±12.5	82.9±8.1	84.7±11.0
NN	86.4±7.8	84.6±13.0	**90.5±3.0**	81.6±15.8	87.4±6.2	82.5±11.8
CNN	84.3±10.9	83.7±17.7	**90.3±4.1**	82.7±16.8	86.1±9.6	82.4±15.6
LSTM	84.3±9.2	84.7±15.0	88.7±3.9	82.9±17.6	87.8±6.7	81.6±15.5
CNN-ST	91.5±3.2	**93.0±2.7**	92.5±2.7	92.5±2.8	91.4±3.2	89.43±7.3

CrisisLexT26	Word2Vec	CrisisW2V	GloVe	CrisisGloVe	FastText	CrisisFastText
SVM-Mean	70.3±14.8	**86.2±3.9**	85.7±4.5	**86.3±4.2**	70.2±14.8	85.5±4.2
NN	83.8±6.0	85.3±4.2	85.0±5.1	85.5±4.4	83.0±5.7	84.8±3.9
CNN	84.5±5.3	**86.3±4.1**	**86.2±4.6**	**86.6±4.1**	84.0±7.1	85.3±5.0
LSTM	84.0±5.4	**86.2±4.0**	85.6±4.9	**86.2±4.6**	84.5±5.4	85.5±4.6
CNN-ST	84.6±4.1	85.6±6.0	85.7±4.1	85.4±3.9	85.2±5.3	85.1±5.1

2CTweet	Word2Vec	CrisisW2V	GloVe	CrisisGloVe	FastText	CrisisFastText
SVM-MMM	89.0±3.7	**92.1±2.5**	90.4±3.7	90.2±3.6	84.0±5.2	89.2±4.0
NN	**92.1±3.3**	91.1±3.3	90.2±3.5	91.0±3.1	91.6±3.5	90.7±3.42
CNN	**92.3±3.4**	91.6±3.8	**92.0±3.4**	91.3±3.5	**92.1±3.7**	90.5±3.2
LSTM	91.7±3.5	91.9±2.9	91.5±3.1	91.5±2.7	91.5±3.3	91.5±3.3
CNN-ST	**92.4±3.3**	91.1±4.3	**92.3±3.6**	91.1±3.8	92.0±4.1	91.7±3.2

similar to the one in Section 5.2: we split the target data into 5 folds; we use 1 fold (20%) as target test data, and the remaining 4 folds (80%) as target unlabeled data.

6.2.2 Results and discussion

The averaged accuracy results of the experiments are shown in Tables 6.6. To compare with traditional models, we also take the rows that contain the best results from Tables 6.2, 6.3 and 6.4, specifically, the results correspondingto the SVM models with Mean or MinMaxMean aggregations of word embeddings. We discuss the results following research questions similat to those in the previous Section 6.1.

experiment with small iteration numbers.Specifically, the results reported here are for three steps. Running more iterations - up to 10 - didn't show a significant improvement over just three iterations.

Among the pre-trained word embeddings and crisis word embeddings, which types work best with different models overall?

From Table 6.6 we can see that the GloVe embeddings pre-trained with general Twitter data, when used with CNN models, can achieve best or close to best results for all three datasets. Overall, GloVe embeddings can get more stable results with deep learning models than with traditional models. While Word2Vec works better for 2Ctweets dataset, CrisisWord2Vec and CrisisGloVe work better for CrisisLexT26. The performance gaps with different word embeddings sometimes are rather large, especially for CrisisLexT6, which also proves the value of evaluating different word embeddings on the given tasks.

As for the benefits of word embeddings trained on crisis tweet corpora, we can find similar pattern as for the experiments performed with traditional supervised models. Word embeddings pre-trained on the general corpus work better for the relevant vs irrelevant tasks, and crisis word embeddings trained on the crisis corpus work best for the informative vs. non-informative task. Specifically, for CrisisLexT26 dataset, CrisisGloVe embeddings and CrisisWord2Vec are better than embeddings pre-trained on the general corpus. Furthermore, for the CrisisLexT6, deep learning models with general pre-trained GloVe embeddings are better than crisis specific embeddings. For the 2CTweets dataset, CNN model with pre-trained general Word2Vec, GloVe and FastText embeddings achieve equivalent results, and all are better than the crisis specific word embeddings overall. The reason for this maybe be similar to the one discussed in Section 6.1.5.

How do the deep learning models compare with traditional models? And which model, between NN, CNN and LSTM, performs best overall for the three datasets?

Comparing deep learning models with SVM models using word embeddings, we can see that the performances of deep learning models is just slightly better and sometimes equivalent to that of the SVM models for all three dataset. For the best deep learning models of a dataset, we can find an SVM model that achieves very close or equivalent results. Therefore, SVM with aggregated word embeddings should be considered as a strong baseline for deep learning models. Among the deep learning models, CNN models achieves the best results overall compared with NN and LSTM models. The reason may be that tweets are short,

Table 6.7: Accuracy results of each disaster in CrisisLexT6 in leave-one-out setting. We represent each target disaster with its initials. The best result for each disaster (column) is highlighted in bold.

	SH	QF	BB	WTE	OT	AF	Averaged
CNN	85.6	94.7	85.8	94.5	**92.2**	89.3	90.3±4.1
CNN-ST-GloVe	**91.5**	94.7	**91.3**	97.3	91.2	89.2	92.5±2.7
CNN-ST-CrisisW2V	90.3	**94.8**	90.0	**97.8**	92.1	**93.2**	**93.0±2.7**

and small local pieces (n-grams) of the tweets can better reflect tweets' labels. CNN models are able to captures different n-grams features and therefore can get better results.

Does self-training help with deep learning models?

Given the best overall word embeddings and the best overall model (CNN), selected based on the results, we want to see whether self-training can further help the CNN models or not. From Table 6.6 we can see, for CrisisLexT6, self-training helps improve the base CNN models for all types embeddings. Using pre-trained Twitter GloVe embeddings and CrisisGloVe, CNN-ST improves the overall average accuracy from about 90% in the CNN model to 92%, which is a 2% improvement. But surprisingly, CNN-ST achieves the best results using crisis embeddings CrisisWord2Vec with the averaged accuracy up to 93%. A closer look into the accuracy results of each disaster in Table 6.7 shows that the improvement of CNN-ST from CNN mainly comes from the better performances on Hurricane Sandy, Boston Bombing, and Albert Flood. Especially Albert Flood, CNN with CrisisWord2Vec has significiant improvements (from 89% to 93%). As the majority of tweets in our crisis corpora are retrieved from hurricanes during the 2017 hurricane season, when several severe floods among cities such as Houston were also reported together with Hurricane Harvey, better results using crisis specific embeddings for Hurricane Sandy and Alberta Flood are expected. However, self-training doesn't help for 2CTweets, and even slightly hurts the performance of the CNN models for CrisisLexT26. The reason maybe that we have enough target unlabeled data in CrisisLexT6 but not for the other two datasets. The size of each dataset is shown in Table 6.1. For CrisisLexT6, for each target disaster/crisis, we have nearly 8,000 tweets, therefore about 6,800 target unlabeled tweets. However, for CrisisLexT26, each

target disaster/crisis has only about 1,000 tweets in total, and some are very unbalanced (e.g., in New York train crash case). For 2CTweets, the size of each target disaster/crisis is more than 1,000, but still less than 3,000. With our leave-one-out setting, the source instances are roughly 10 times the target unlabeled instances, therefore still dominate the models' performances. We believe self-training should, in general, help deep learning models when we have a very large number of target unlabeled data. But the training cost increases significantly also comparing with just train the model once. Furthermore, the training cost is even higher with larger dataset. For our experiments, the good outcome is that the results of ten iterations doesn't change much compared to the outcome with three iterations.

6.3 Pre-trained language model with multi-source domain adversarial networks

In this section, we experiment with pre-trained language based models, specifically models based on BERT (Devlin et al., 2018). We take the results of CNN and CNN-ST models from the previous section as baselines, and run BERT based models. We also run the multi-source domain adversarial networks model (MDAN) (Zhao et al., 2017) with averaged word embedding representations, and finally representations using BERT language model outputs.

6.3.1 Experimental setup

As a pre-trained language model trained on large datasets, BERT can be used here in different ways. We can simply take the final hidden vector of the first input token [cls], add one classification layer and fine-tune the model with sources labeled data, which is usually referred as default BERT model. We can also just use embeddings computed from a pre-trained BERT language model to get token representations, and use them in other models just like we use pre-trained word embeddings in traditional supervised models. This use of BERT with other models can be seen as a feature representation based approach. Here, we experiment with both fine-tuning and the embeddings idea. For MDAN models, the

original paper is using 5000 dimensional feature vectors of unigram and bigram tokens, and three hidden layers in the networks (with 5000, 500, 100 units, respectively). We will use averaged word embeddings as representations, as well as average of the final hidden state sequences of BERT pre-trained language model. BERT has two pre-trained models: base and large. We will use the base one, specifically "bert-base-uncased", which has 12 layers of transformer blocks and 12 attention heads, while the hidden dimension is 768. The details of the experiments are as follows:

- BERT: This model adds a fully connect layer to the language model. It takes the final hidden vector of the first input token (a 768 dimension vector) as the aggregation of a tweet, and passes it to the classification layer. The whole model is fine-tuned with sources labeled instances.

- BERT-CNN: Instead of just using the hidden vector of the first input token, here we will use the hidden states of all layers of BERT. The sequence of the hidden states of each layer is passed to a convolutional layer, followed by a fully connected layer as the classification layer. The model is set as in (Ma, 2019), where the convolutional layer has 16 filters of size 3,768 and the out channels are then concatenated together (16 filters, 12 layers, 192 in total).

- BERT-LSTM: With this model, the final hidden states sequence of all tokens in a tweet are used, and the sequence is passed to a 256-units bidirectional LSTM layer followed by a classification layer.

- MDAN: For this model, we use averaged pre-trained word embeddings as representations, specifically, the best representations based on the previous section results. We use 100-dimensional pre-trained general Twitter GloVe embeddings for CrisisLexT6, 100-dimensional crisis specific word embeddings, CrisisGloVe, for CrisisLexT26, and 300-dimensional pre-trained Word2Vec word embeddings for 2CTweets. The units of the three hidden layers are (word embedding dimension, 64, 32).

- MDAN-BERT: For this model, we take the final hidden states of all tokens of a tweet

Table 6.8: Averaged accuracy results of three datasets with BERT and MDAN based models.

	CrisisLexT6	CrisisLexT26	2CTweets
CNN	90.3±4.1	86.6±4.1	92.3±3.4
CNN-ST	**93.0±2.7**	85.4±3.9	92.4±3.3
BERT	88.3±7.4	**87.7±4.1**	**93.1±3.1**
BERT-CNN	88.8±6.6	**87.6±4.1**	92.9±3.4
BERT-LSTM	88.9±6.7	**87.5±4.2**	92.9±3.0
MDAN	90.2±3.7	86.1±4.3	90.4±3.5
MDAN-BERT	89.1±4.4	**87.2±3.3**	91.4±3.5

and average them to one vector as the representation of that tweet. Thus, each tweet is still represented as a 768-dimensional vector like in BERT default model, but the average on all token final hidden states usually works better than the vector of the first input token. Then we feed the representations from BERT to a MDAN model that has hidden units (768, 256, 64).

We still use the same leave-one-out setting as in two previous sections.

6.3.2 Results and discussion

The average accuracy results of the experiments are shown in Table 6.8. The best values of each dataset (column) is highlighted in bold. Comparing with CNN models, BERT based models work better for CrisisLexT26, sightly better than CNN baselines for 2CTweets, but not better than the CNN model, especially the CNN-ST model for CrisisLexT6. We suspect that too many source instances may deteriorate the model performance on the target domain, especially with the fine-tuning approach. To investigate this further, for CrisisLexT6, we did a simple experiment using just 10% percent of all available sources instance for training, and the results show that we can achieve equivalent averaged accuracy as the CNN model. This suggests that we may be able to get better results using carefully selected source instances with BERT fine-tuned models. If we compare BERT-CNN and BERT-LSTM with the BERT default fine-tuned model, the results are very close to each other. For 2CTweets, the BERT default model is actually better, although the differences are not significant. This suggests

that we may not need to use complex layers on top of BERT, as a simple layer with fine-tuning can work well.

Similarly, for the two MDAN models, MDAN-BERT model slightly improves the performance of the CNN baseline for CrisisLexT26, but not for the other two datasets for which the results are actually worse than those of the CNN models. The results of the MDAN models for CrisisLexT6 and CrisisLexT26 are close or equivalent to CNN baselines, but are worse than the CNN baselines for 2CTweets. Our preliminary experiments with hyper-parameter tuning for the MDAN models also show that these models are not highly sensitive to the hyper-parameters, meaning the performance of the model may not improve when changing some hyper-parameters, although more careful hyper-parameter tuning is needed to finally conclude that. But from the representation comparison perspective, the MDAN model works better with BERT representations than with the word embedding representations for two datasets, CrisisLextT26 and 2CTweets. This is an intuitive results, considering that pre-trained language models are built with much more deep models than the models used to train word embeddings.

To further investigate where the differences between model performances are observed, we look into the accuracy results for each disaster in CrisisLexT6 as in Table 6.9. We take CNN as the baseline, and compare BERT, MDAN-embeddings and CNN-ST for each target disaster in CrisisLexT6. Recall that the experimental setting is leave-one-out, where one disaster is taken as target, and the other disasters in the dataset are included in the sources. It can be observed that, when compared with the CNN baseline, CNN-ST can significantly improve the performance on three disasters, SH, BB and AF, it performs slightly worse on one disaster (OT), and it does not deteriorate the performance on the other two (QF, WTE). Compared with the CNN baseline, BERT model improves the performance significantly for only one target (QF), slightly improves the performance for two disasters (WTE, OT), and performs much worse than the baseline for three disasters (SH, BB, AF). Interestingly, the BERT model behaves almost the opposite of the CNN-ST. The BERT model performs better than the CNN baseline, where the CNN-ST performs worse, for example QF and OT, while the BERT model performs much worse than the baseline when CNN-ST performs better,

Table 6.9: Accuracy results of each disaster in CrisisLexT6 in leave-one-out setting. We represent each target disaster with its initials. The best result for each disaster (column) is highlighted in bold.

	SH	QF	BB	WTE	OT	AF	Averaged
CNN	85.6	94.7	85.8	94.5	92.2	89.3	90.3±4.1
BERT	79.4	**96.0**	82.0	95.8	**92.6**	83.9	88.3±7.4
MDAN	85.6	92.8	89.1	95.1	91.7	86.8	90.2±3.7
CNN-ST	**90.3**	94.8	**90.0**	**97.8**	92.1	**93.2**	**93.0±2.7**

specifically, for SH and BB. Fine-tuning a pre-trained language model can easily overfit the sources, therefore the performance may deteriorate when the target disaster is too different from the sources, as in the case of Hurricane Sandy (SH). The MDAN model performs better than the CNN baseline for two disasters, BB and WET, and worse for three disasters, QF, OT and AF. If we only compare disasters of the same type, it's interesting to see that the MDAN model can improve the results for man-made disasters, Boston Bombing (BB) and West Texas Explosion (WTE) but not for floods (QF and AF). The reason may be that the distances between the distribution of BB and the distributions of the other source disasters, such as SH or OT, are bigger than the distance between the distribution of QF and the distributions of its source disasters, e.g., SH or OT, because both a Hurricane and a Tornado may cause floods as well. The adversarial learning of the MDAN models may improve the results more when the target and sources domains are further apart.

6.4 Conclusions

In this chapter, we experimented with embeddings based representation approaches, which utilized word embeddings, sentence encodings, and pre-trained language models, in a multi-source domain adaptation setting. We first evaluated how different representations using pre-trained word embeddings and sentence encodings perform with supervised learning models in Section 6.1. We then evaluated pre-trained word embedding with deep learning models in Section 6.2, and also proposed to run self-training with a CNN model here. Finally, in Section

6.3, we used the state-of-the-art pre-trained language model BERT on the classification tasks and finally applied a multi-source domain adversarial model using the representations obtained from the pre-trained BERT model.

In Section 6.1, we used the bag-of-word representations as baselines, and compared simple feature or representation-based approaches with pre-trained word embeddings. We found that this simple feature based adaptation could help improve the performances. We then compared three types of word embeddings (Word2Vec, GloVe and FastText) and three approaches to aggregate the word embeddings into tweet representations (specifically, Mean, MinMaxMean, and TfidfMean). We performed an extrinsic evaluation, where the embeddings were used with four traditional machine learning algorithms (Naive Bayes, Random Forest, K-Nearest Neighbors and Support Vector Machines) for crisis tweet classification tasks.

We used both word embeddings pre-trained on corpora such as Google News, Wikipedia or Twitter, and crisis-specific embeddings trained on our tweet corpora. We observed that the crisis-specific embeddings were more suitable for more specific crisis-related tasks, while the pre-trained embeddings are more suitable for more general classification tasks, where not all the tweets classified are crisis related. Among the three types of word embeddings (Word2Vec, GloVe and FastText), the GloVe embeddings performed the best overall on the three datasets used in our study (CrisisLexT6, CrisisLexT26 and 2CTweets). Specifically, the pre-trained GloVe embeddings worked better for the more general classification task in CrisisLexT6, while the CrisisGloVe performed better for the more specific classification tasks in CrisisLexT26. Furthermore, among the traditional supervised models, SVM models were shown to make the best use of the embeddings in terms of generalizing to data from future disasters.

In addition to word embeddings, we also experimented with models for sentence encoding in Section 6.1, and showed that the sentence encoders have great potential for being used in crisis tweet classification tasks. For example, the sentence encoding representations worked much better than the aggregated pre-trained word embeddings for the Naive Bayes classifier. This suggests that we can get better representations with sentence encoding models than

just aggregating pre-trained word embeddings.

In Section 6.2, an extrinsic evaluation of the different types of word embedding was performed with deep learning models, namely Neural networks (NN), Convolutional neural networks (CNN) and Long short-term memory networks (LSTM). We found similar patterns regarding the types of word embeddings that work better. CNN models, in general, worked better than the other two types of models. Given that, we further ran self-training with CNN as base model, and improved the averaged accuracy of the CNN model by up to 3% for CrisisLexT6. Therefore, we claim that self-training holds great potential if a larger number of target unlabeled data is available.

In Section 6.3, we ran BERT fine-tuned classification models, and MDAN models with representations from pre-trained word embeddings and from pre-trained BERT language models. We found that BERT-based models, including fine-tuned models, and MDAN with the BERT representation model (MDAN-BERT) were better than the CNN baselines for CrisisLexT26, but not for the other two datasets. Multi-domain differences may be one reason that BERT based models, as well as MDAN models have worse performance for the CrisisLexT6 and 2CTweets datasets.

To conclude, our study provides insights into to how to build "generalized" classifiers that are ready to use in real crisis situations. For an emergent disaster, the first classification task to be addressed is the classification of tweets as relevant or irrelevant to the disaster (similar to CrisisLexT6). Subsequently, relevant tweets can be classified as informative and non-informative (CrisisLexT26), and further into specific situational awareness categories, such as injured people, damaged infrastructure, etc. Our preliminary results suggest that the relevant versus irrelevant classification task may be addressed using a "generalized" classifier that employs Twitter pre-trained GloVe embeddings with a CNN model, and if time allows, we can use self-training to improve the classifier's performance for the specific target disaster. The informative versus non-informative classification task may be addressed with a "generalized" classifier that employs a pre-trained language model (BERT) and the fine-tuning approach, assuming that we can get labeled source disaster data that cover different types of disasters or crises.

Chapter 7

Conclusions and future directions

In this thesis, we have studied domain adaptation problems for social media crisis data classification tasks, specifically crisis tweet classification. We proposed simple yet effective domain adaptation approaches that could be used in practice in disaster management. Concretely, the main contributions are summarized in what follows.

In Chapter 5, we proposed domain adaptation approaches based on Naive Bayes with Expectation Maximization (EM) and self-training strategies. We also performed a hyper-parameter analysis to provide guidance for potential practical application. We further compared the self-training approach with a feature representation based approach, Correlation Alignment (CORAL), used on top of supervised Naive Bayes classifiers, and proposed a hybrid model combining self-training and CORAL on top of Naive Bayes models. We experimented with different source and target disaster pairs on predicting the disaster related tweets.

In Chapter 6, we utilized word embeddings, sentence encodings, and pre-trained language models in a multi-source domain adaptation setting. We first evaluated how different representations using pre-trained word embeddings and sentence encoding models perform with supervised learning models in Section 6.1. We then evaluated pre-trained word embedding with deep learning models in Section 6.2, we also proposed to run self-training with a CNN model here. Finally, in Section 6.3, we experimented with approaches based on state-of-

the-art pre-trained language models, specifically BERT, and applied a multi-source domain adversarial model using the representations computed from pre-trained BERT model.

For parameter and instance based domain adaptation approaches, we found that EM/self-training represent effective approaches for domain adaptation problems if we had large target unlabeled data. For feature/representation based approaches, simple approaches which aggregate pre-trained word embeddings could improve the performance of supervised models with bag-of-words representations in the supervised settings. The extrinsic evaluation of different pre-trained word embeddings showed that such evaluation was needed for the classification tasks. For our datasets, GloVe word embeddings worked better overall. For the pre-trained language model based approaches, we found that smaller datasets benefited more from pre-trained language model (BERT) with the fine-tuning approach. This approach has great potential and needs further investigation.

As for future work, there are still many challenges and problems that need to be addressed From the social media crisis data point of view, we need more diverse data that will cover more types of disasters/crises. Multi-modal data that includes not only texts but also images, or even videos, are also needed to perform more insightful analyses. From the classification tasks point of view, we can extend our approaches to more classification tasks, such as multi-class classification tasks for situational awareness categories. It would also be useful to consider the classification of eyewitness tweets versus non-eyewitness tweets. Many other classification tasks that are important for emergency management are also challenging and need more research efforts, such as misinforamtion/disinformation labeling, cross-language classifications task, and event/object/location detection, as well as damage evaluation, among others. From the approaches perspective, we can experiment with more instance weight and selection approaches, especially in the multi-source domain adaptation setting. We can collect a large disaster/crisis dataset, for example, from Wikipedia, and use that to train a language model that is specific for disaster/crisis, and then fine-tune the model for downstream crisis data classification tasks. We have experimented with single-source domain adaptation and multi-source domain adaptation approaches, with focus on unsupervised domain adaptation. Other transfer learning approaches, like multi-task learn-

ing, would be interesting too when we have small amount of target labeled data. Similarly, cross-lingual models are of interest, to help crisis management in countries where they have more limited resources in terms of model training. In addition, zero-shot or few-shot learning can be valuable too for cases where we do not have disaster/crisis labeled tweets for all classes. Finally, domain adaptation approaches for multi-modal social media data that have both texts and visual data, such as images or videos, could also be of great value to enhance situational awareness in disaster management.

Bibliography

Using Social Media for Enhanced Situation Awareness and Decision Support, 2014.

H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand. Domain-adversarial neural networks. *CoRR*, abs/1412.4446, 2014. URL http://arxiv.org/abs/1412.4446.

F. Alam, S. R. Joty, and M. Imran. Domain adaptation with adversarial training and graph embeddings. In I. Gurevych and Y. Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1077–1087. Association for Computational Linguistics, 2018a. doi: 10.18653/v1/P18-1099. URL https://www.aclweb.org/anthology/P18-1099/.

F. Alam, F. Ofli, and M. Imran. Crisismmd: Multimodal twitter datasets from natural disasters. In *Proceedings of the 12th International AAAI Conference on Web and Social Media (ICWSM)*, June 2018b.

S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SyK00v5xx.

Z. Ashktorab, C. Brown, M. Nandi, and A. Culotta. Tweedr: Mining twitter to inform disaster response. In S. R. Hiltz, L. Plotnick, M. Pfaf, and P. C. Shih, editors, *11th Proceedings of the International Conference on Information Systems for Crisis Response and Management, University Park, Pennsylvania, USA, May 18-21, 2014*. ISCRAM Association, 2014. URL http://idl.iscram.org/files/ashktorab/2014/275_Ashktorab_etal2014.pdf.

M. Baroni, G. Dinu, and G. Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 238–247, 2014.

G. Beigi, X. Hu, R. Maciejewski, and H. Liu. An overview of sentiment analysis in social media and its applications in disaster relief. In *Sentiment Analysis and Ontology Engineering. Springer International Publishing*, pages 313—-340, 2016.

S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems 20*, Cambridge, MA, 2007. MIT Press.

J. Blitzer, R. T. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In D. Jurafsky and É. Gaussier, editors, *EMNLP 2006, Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, 22-23 July 2006, Sydney, Australia*, pages 120–128. ACL, 2006. URL https://www.aclweb.org/anthology/W06-1615/.

J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics*, Prague, Czech Republic, 2007.

J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems 21*, Cambridge, MA, 2008. MIT Press.

A. Blum and T. M. Mitchell. Combining labeled and unlabeled data with co-training. In P. L. Bartlett and Y. Mansour, editors, *Proceedings of the Eleventh Annual Conference on Computational Learning Theory, COLT 1998, Madison, Wisconsin, USA, July 24-26, 1998*, pages 92–100. ACM, 1998. doi: 10.1145/279943.279962. URL https://doi.org/10.1145/279943.279962.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. ISSN 2307-387X.

C. D. Boom, S. V. Canneyt, T. Demeester, and B. Dhoedt. Representation learning for very short texts using weighted word embedding aggregation. *CoRR*, abs/1607.00570, 2016. URL http://arxiv.org/abs/1607.00570.

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992*, pages 144–152. ACM, 1992. doi: 10.1145/130385.130401. URL https://doi.org/10.1145/130385.130401.

L. Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, 1996. doi: 10.1007/BF00058655. URL https://doi.org/10.1007/BF00058655.

L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. ISBN 0-534-98053-8.

C. Caragea, A. C. Squicciarini, S. Stehle, K. Neppalli, and A. H. Tapia. Mapping moods: Geo-mapped sentiment analysis during hurricane sandy. In S. R. Hiltz, L. Plotnick, M. Pfaf, and P. C. Shih, editors, *11th Proceedings of the International Conference on Information Systems for Crisis Response and Management, University Park, Pennsylvania, USA, May 18-21, 2014*. ISCRAM Association, 2014. URL http://idl.iscram.org/files/caragea/2014/372_Caragea_etal2014.pdf.

C. Caragea, A. Silvescu, and A. H. Tapia. Identifying informative messages in disasters using convolutional neural networks. In A. H. Tapia, P. Antunes, V. A. Bañuls, K. A. Moore, and J. P. de Albuquerque, editors, *13th Proceedings of the International Conference*

on Information Systems for Crisis Response and Management, Rio de Janeiro, Brasil, May 22-25, 2016. ISCRAM Association, 2016. URL http://idl.iscram.org/files/corneliacaragea/2016/1397_CorneliaCaragea_etal2016.pdf.

C. Castillo. *Big Crisis Data: Social Media in Disasters and Time-Critical Situations*. Cambridge University Press, 2016.

D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y. Sung, B. Strope, and R. Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018. URL http://arxiv.org/abs/1803.11175.

C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM TIST*, 2(3): 27:1–27:27, 2011. doi: 10.1145/1961189.1961199. URL http://doi.acm.org/10.1145/1961189.1961199.

M. Chen, K. Q. Weinberger, and J. Blitzer. Co-training for domain adaptation. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 2456–2464, 2011a. URL http://papers.nips.cc/paper/4433-co-training-for-domain-adaptation.

M. Chen, K. Q. Weinberger, and Y. Chen. Automatic feature decomposition for single view co-training. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 953–960. Omnipress, 2011b. URL https://icml.cc/2011/papers/498_icmlpaper.pdf.

M. Chen, Z. E. Xu, K. Q. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. *CoRR*, abs/1206.4683, 2012. URL http://arxiv.org/abs/1206.4683.

J. R. Chowdhury, C. Caragea, and D. Caragea. Keyphrase extraction from disaster-related tweets. In L. Liu, R. W. White, A. Mantrach, F. Silvestri, J. J. McAuley, R. Baeza-Yates, and L. Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 1555–1566. ACM, 2019. doi: 10.1145/3308558.3313696. URL https://doi.org/10.1145/3308558.3313696.

J. R. Chowdhury, C. Caragea, and D. Caragea. On identifying hashtags in disaster twitter data. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 498–506. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/5387.

A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364, 2017. URL http://arxiv.org/abs/1705.02364.

C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995. doi: 10.1007/BF00994018. URL https://doi.org/10.1007/BF00994018.

A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3079–3087, 2015. URL http://papers.nips.cc/paper/5949-semi-supervised-sequence-learning.

W. Dai, G. Xue, Q. Yang, and Y. Yu. Transferring naive bayes classifiers for text classification. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 540–545. AAAI Press, 2007. URL http://www.aaai.org/Library/AAAI/2007/aaai07-085.php.

H. Daumé, III, A. Kumar, and A. Saha. Frustratingly easy semi-supervised domain adapta-

tion. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, DANLP 2010, pages 53–59, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 978-1-932432-80-0. URL http://dl.acm.org/citation.cfm?id=1870526.1870534.

H. Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263. Association for Computational Linguistics, 2007. URL http://www.aclweb.org/anthology/P07-1033.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

S. Desai, C. Caragea, and J. J. Li. Detecting perceived emotions in hurricane disasters. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5290–5305. Association for Computational Linguistics, 2020. URL https://www.aclweb.org/anthology/2020.acl-main.471/.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

C. Du, H. Sun, J. Wang, Q. Qi, and J. Liao. Adversarial and domain-aware BERT for cross-domain sentiment analysis. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4019–4028. Association for Computational Linguistics, 2020. URL https://www.aclweb.org/anthology/2020.acl-main.370/.

L. Duan, I. W. Tsang, D. Xu, and T. Chua. Domain adaptation from multiple sources via auxiliary classifiers. In A. P. Danyluk, L. Bottou, and M. L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009,*

Montreal, Quebec, Canada, June 14-18, 2009, volume 382 of *ACM International Conference Proceeding Series*, pages 289–296. ACM, 2009. doi: 10.1145/1553374.1553411. URL https://doi.org/10.1145/1553374.1553411.

J. L. Elman. Finding structure in time. *Cogn. Sci.*, 14(2):179–211, 1990. doi: 10.1207/s15516709cog1402_1. URL https://doi.org/10.1207/s15516709cog1402_1.

C. Fan, F. Wu, and A. Mostafavi. A hybrid machine learning pipeline for automated mapping of events and locations from social media in disasters. *IEEE Access*, 8:10478–10490, 2020. doi: 10.1109/ACCESS.2020.2965550. URL https://doi.org/10.1109/ACCESS.2020.2965550.

R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. doi: 10.1145/1390681.1442794. URL http://doi.acm.org/10.1145/1390681.1442794.

M. Faruqui and C. Dyer. Community evaluation and exchange of word vectors at wordvectors. org. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 19–24, 2014.

Y. Ganin and V. S. Lempitsky. Unsupervised domain adaptation by backpropagation. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1180–1189. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/ganin15.html.

Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. S. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17:59:1–59:35, 2016. URL http://jmlr.org/papers/v17/15-239.html.

H. Gao, G. Barbier, and R. Goolsby. Harnessing the crowdsourcing power of social media for

disaster relief. *IEEE Intelligent Systems*, 26(3):10–14, 2011. doi: 10.1109/MIS.2011.52. URL https://doi.org/10.1109/MIS.2011.52.

F. A. Gers, J. A. Schmidhuber, and F. A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Comput.*, 12(10):2451–2471, Oct. 2000. ISSN 0899-7667. doi: 10.1162/089976600300015015. URL https://doi.org/10.1162/089976600300015015.

X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 513–520, USA, 2011. Omnipress. ISBN 978-1-4503-0619-5. URL http://dl.acm.org/citation.cfm?id=3104482.3104547.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.

I. J. Goodfellow, Y. Bengio, and A. C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016. ISBN 978-0-262-03561-3. URL http://www.deeplearningbook.org/.

J. Guo, D. J. Shah, and R. Barzilay. Multi-source domain adaptation with mixture of experts. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4694–4703. Association for Computational Linguistics, 2018. URL https://www.aclweb.org/anthology/D18-1498/.

Y. Guo, H. Zhang, and B. Spencer. Cost-sensitive self-training. In *Proceedings of the 25th Canadian Conference on Advances in Artificial Intelligence*, Canadian AI'12, pages 74–

84, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-30352-4. doi: 10.1007/978-3-642-30353-1_7. URL http://dx.doi.org/10.1007/978-3-642-30353-1_7.

M. A. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009. doi: 10.1145/1656274.1656278. URL http://doi.acm.org/10.1145/1656274.1656278.

W. L. Hamilton, K. Clark, J. Leskovec, and D. Jurafsky. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, Austin, Texas, Nov. 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1057. URL https://www.aclweb.org/anthology/D16-1057.

N. Herndon and D. Caragea. An evaluation of self-training styles for domain adaptation on the task of splice site prediction. In J. Pei, F. Silvestri, and J. Tang, editors, *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015, Paris, France, August 25 - 28, 2015*, pages 1042–1047. ACM, 2015. doi: 10.1145/2808797.2808809. URL https://doi.org/10.1145/2808797.2808809.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, Nov. 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

J. Hoffman, B. Kulis, T. Darrell, and K. Saenko. Discovering latent domains for multisource domain adaptation. In A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part II*, volume 7573 of *Lecture Notes in Computer Science*, pages 702–715. Springer, 2012. doi: 10.1007/978-3-642-33709-3_50. URL https://doi.org/10.1007/978-3-642-33709-3_50.

J. Hoffman, E. Rodner, J. Donahue, B. Kulis, and K. Saenko. Asymmetric and category in-

variant feature transformations for domain adaptation. *International Journal in Computer Vision (IJCV)*, 2013.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. ISSN 0027-8424. doi: 10.1073/pnas.79.8.2554. URL https://www.pnas.org/content/79/8/2554.

J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In I. Gurevych and Y. Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1031. URL https://www.aclweb.org/anthology/P18-1031/.

J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf. Correcting sample selection bias by unlabeled data. In B. Schölkopf, J. C. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 601–608. MIT Press, 2006. URL http://papers.nips.cc/paper/3075-correcting-sample-selection-bias-by-unlabeled-data.

A. L. Hughes, L. A. S. Denis, L. Palen, and K. M. Anderson. Online public communications by police & fire services during the 2012 hurricane sandy. In M. Jones, P. A. Palanque, A. Schmidt, and T. Grossman, editors, *CHI Conference on Human Factors in Computing Systems, CHI'14, Toronto, ON, Canada - April 26 - May 01, 2014*, pages 1505–1514. ACM, 2014. doi: 10.1145/2556288.2557227. URL https://doi.org/10.1145/2556288.2557227.

M. Imran, S. Elbassuoni, C. Castillo, F. Diaz, and P. Meier. Practical extraction of disaster-relevant information from social media. In L. Carr, A. H. F. Laender, B. F. Lóscio, I. King, M. Fontoura, D. Vrandecic, L. Aroyo, J. P. M. de Oliveira, F. Lima, and E. Wilde, editors, *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro,*

Brazil, May 13-17, 2013, Companion Volume, pages 1021–1024. International World Wide Web Conferences Steering Committee / ACM, 2013a. ISBN 978-1-4503-2038-2. URL http://dl.acm.org/citation.cfm?id=2488109.

M. Imran, S. Elbassuoni, C. Castillo, F. Diaz, and P. Meier. Extracting information nuggets from disaster- related messages in social media. In T. Comes, F. Fiedrich, S. Fortier, J. Geldermann, and T. Müller, editors, *10th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Baden-Baden, Germany, May 12-15, 2013*. ISCRAM Association, 2013b. ISBN 978-3-923704-80-4. URL http://idl.iscram.org/files/imran/2013/613_Imran_etal2013.pdf.

M. Imran, C. Castillo, F. Diaz, and S. Vieweg. Processing social media messages in mass emergency: A survey. *ACM Comput. Surv.*, 47(4):67:1–67:38, 2015. doi: 10.1145/2771588. URL http://doi.acm.org/10.1145/2771588.

M. Imran, P. Mitra, and J. Srivastava. Cross-language domain adaptation for classifying crisis-related short messages. In A. H. Tapia, P. Antunes, V. A. Bañuls, K. A. Moore, and J. P. de Albuquerque, editors, *13th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Rio de Janeiro, Brasil, May 22-25, 2016*. ISCRAM Association, 2016. URL http://idl.iscram.org/files/muhammadimran/2016/1396_MuhammadImran_etal2016.pdf.

J. Jiang. A literature survey on domain adaptation of statistical classifiers. *URL: http://sifaka. cs. uiuc. edu/jiang4/domainadaptation/survey*, 2008.

J. Jiang and C. Zhai. A two-stage approach to domain adaptation for statistical classifiers. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 401–410, New York, NY, USA, 2007a. ACM. ISBN 978-1-59593-803-9. doi: 10.1145/1321440.1321498. URL http://doi.acm.org/10.1145/1321440.1321498.

J. Jiang and C. Zhai. Instance weighting for domain adaptation in nlp. In *Proceedings of*

the 45th Annual Meeting of the Association of Computational Linguistics, pages 264–271, Prague, Czech Republic, June 2007b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P07-1034.

M. Jordan. Serial order: a parallel distributed processing approach. technical report, june 1985-march 1986. Technical report, California Univ., San Diego, La Jolla (USA). Inst. for Cognitive Science, 1986.

P. Khare. *Identifying and Processing Crisis Information from Social Media*. PhD thesis, The Open University, February 2020. URL http://oro.open.ac.uk/69594/.

Y. Kim. Convolutional neural networks for sentence classification. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751. ACL, 2014. doi: 10.3115/v1/d14-1181. URL https://doi.org/10.3115/v1/d14-1181.

S. Kumar, X. Hu, and H. Liu. A behavior analytics approach to identifying tweets from crisis regions. In L. Ferres, G. Rossi, V. A. F. Almeida, and E. Herder, editors, *25th ACM Conference on Hypertext and Social Media, HT '14, Santiago, Chile, September 1-4, 2014*, pages 255–260. ACM, 2014. doi: 10.1145/2631775.2631814. URL https://doi.org/10.1145/2631775.2631814.

P. M. Landwehr and K. M. Carley. *Social Media in Disaster Relief*, pages 225–257. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-642-40837-3. doi: 10.1007/978-3-642-40837-3_7. URL http://dx.doi.org/10.1007/978-3-642-40837-3_7.

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

H. Li and D. Caragea. Crisis tweet classification with pre-trained language model. In *In Preparation*, 2020a.

H. Li, N. Guevara, N. Herndon, D. Caragea, K. Neppalli, C. Caragea, A. C. Squicciarini, and A. H. Tapia. Twitter mining for disaster response: A domain adaptation approach. In L. Palen, M. Büscher, T. Comes, and A. L. Hughes, editors, *12th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Krystiansand, Norway, May 24-27, 2015*. ISCRAM Association, 2015. URL http://idl.iscram.org/files/hongminli/2015/1234_HongminLi_etal2015.pdf.

H. Li, D. Caragea, and C. Caragea. Towards practical usage of a domain adaptation algorithm in the early hours of a disaster. In T. Comes, F. Bénaben, C. Hanachi, M. Lauras, and A. Montarnal, editors, *14th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Albi, France, May 21-24, 2017*. ISCRAM Association, 2017a. URL http://idl.iscram.org/files/hongminli/2017/1503_HongminLi_etal2017.pdf.

H. Li, D. Caragea, C. Caragea, and N. Herndon. Disaster response aided by tweet classification with a domain adaptation approach. *Journal of Contingencies and Crisis Management*, 26(1):16–27, 2018a.

H. Li, X. Li, D. Caragea, and C. Caragea. Comparison of word embeddings and sentence encodings as generalized representations for crisis tweet classification tasks. In K. Stock and D. Bunker, editors, *Proceedings of ISCRAM Asia Pacific 2018: Innovating for Resilience – 1st International Conference on Information Systems for Crisis Response and Management Asia Pacific.*, pages 480–493, 2018b. URL http://idl.iscram.org/files/hongminli/2018/1689_HongminLi_etal2018.pdf.

H. Li, O. Sopova, D. Caragea, and C. Caragea. Domain adaptation for crisis data using correlation alignment and self-training. *Int. J. Inf. Syst. Crisis Response Manag.*, 10(4): 1–20, 2018c. doi: 10.4018/IJISCRAM.2018100101. URL https://doi.org/10.4018/IJISCRAM.2018100101.

X. Li and D. Caragea. Domain adaptation with reconstruction for disaster tweet classification. In J. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. Wen, and

Y. Liu, editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 1561–1564. ACM, 2020b. doi: 10.1145/3397271.3401242. URL https://doi.org/10.1145/3397271.3401242.

X. Li, D. Caragea, C. Caragea, M. Imran, and F. Ofli. Identifying disaster damage images using a domain adaptation approach. In Z. Franco, J. J. González, and J. H. Canós, editors, *Proceedings of the 16th International Conference on Information Systems for Crisis Response and Management, València, Spain, May 19-22, 2019*. ISCRAM Association, 2019. URL http://idl.iscram.org/files/xukunli/2019/1853_XukunLi_etal2019.pdf.

Z. Li, Y. Zhang, Y. Wei, Y. Wu, and Q. Yang. End-to-end adversarial memory network for cross-domain sentiment classification. In C. Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2237–2243. ijcai.org, 2017b. doi: 10.24963/ijcai.2017/311. URL https://doi.org/10.24963/ijcai.2017/311.

G. Ma. Tweets classification with bert in the field of disaster management. 2019.

C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation with multiple sources. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1041–1048. Curran Associates, Inc., 2008. URL http://papers.nips.cc/paper/3550-domain-adaptation-with-multiple-sources.

R. Mazloom, H. Li, D. Caragea, M. Imran, and C. Caragea. Classification of twitter disaster data using a hybrid feature-instance adaptation approach. In K. Boersma and B. M. Tomaszewski, editors, *Proceedings of the 15th International Conference on Information*

Systems for Crisis Response and Management, Rochester, NY, USA, May 20-23, 2018. ISCRAM Association, 2018. URL http://idl.iscram.org/files/rezamazloom/2018/1594_RezaMazloom_etal2018.pdf.

R. Mazloom, H. Li, D. Caragea, C. Caragea, and M. Imran. A hybrid domain adaptation approach for identifying crisis-relevant tweets. *Int. J. Inf. Syst. Crisis Response Manag.*, 11 (2):1–19, 2019. doi: 10.4018/IJISCRAM.2019070101. URL https://doi.org/10.4018/IJISCRAM.2019070101.

P. Meier. Digital humanitarians: how big data is changing the face of humanitarian response. Crc Press, 2015.

M. Mendoza, B. Poblete, and C. Castillo. Twitter under crisis: can we trust what we rt? In C. L. Giles, P. Mitra, I. Perisic, J. Yen, and H. Zhang, editors, *Proceedings of the 3rd Workshop on Social Network Mining and Analysis, SNAKDD 2009, Paris, France, June 28, 2009*, pages 71–79. ACM, 2010. doi: 10.1145/1964858.1964869. URL https://doi.org/10.1145/1964858.1964869.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013a. URL http://arxiv.org/abs/1301.3781.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013b. URL http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.

T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. ISBN 0070428077, 9780070428072.

A. Mohan, Z. Chen, and K. Weinberger. Web-search ranking with initialized gradient boosted regression trees. volume 14 of *Proceedings of Machine Learning Research*, pages 77–89, Haifa, Israel, 25 Jun 2011. PMLR. URL http://proceedings.mlr.press/v14/mohan11a.html.

National Research Council. *Public Response to Alerts and Warnings Using Social Media: Report of a Workshop on Current Knowledge and Research Gaps*. The National Academies Press, Washington, DC, 2013. ISBN 978-0-309-29033-3. doi: 10.17226/15853. URL https://www.nap.edu/catalog/15853/public-response-to-alerts-and-warnings-using-social-media-report.

N. Nayak, G. Angeli, and C. D. Manning. Evaluating word embeddings using a representative suite of practical tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 19–23, 2016.

D. Q. Nguyen, T. Vu, and A. T. Nguyen. Bertweet: A pre-trained language model for english tweets, 2020.

D. T. Nguyen, K. Al-Mannai, S. R. Joty, H. Sajjad, M. Imran, and P. Mitra. Rapid classification of crisis-related data on social networks using convolutional neural networks. *CoRR*, abs/1608.03902, 2016. URL http://arxiv.org/abs/1608.03902.

K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the 2000 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, November 6-11, 2000*, pages 86–93. ACM, 2000. doi: 10.1145/354756.354805. URL http://doi.acm.org/10.1145/354756.354805.

K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2):103–134, May 2000.

ISSN 1573-0565. doi: 10.1023/A:1007692713085. URL https://doi.org/10.1023/A:1007692713085.

A. Olteanu, C. Castillo, F. Diaz, and S. Vieweg. Crisislex: A lexicon for collecting and filtering microblogged communications in crises. In E. Adar, P. Resnick, M. D. Choudhury, B. Hogan, and A. H. Oh, editors, *Proceedings of the Eighth International Conference on Weblogs and Social Media, ICWSM 2014, Ann Arbor, Michigan, USA, June 1-4, 2014*. The AAAI Press, 2014. URL http://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8091.

A. Olteanu, S. Vieweg, and C. Castillo. What to expect when the unexpected happens: Social media communications across crises. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, pages 994–1009, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-2922-4. doi: 10.1145/2675133.2675242. URL http://doi.acm.org/10.1145/2675133.2675242.

L. Palen and K. M. Anderson. Crisis informatics-new data for extraordinary times. *Science*, 353(6296):224–225, 2016.

L. Palen and A. L. Hughes. *Social Media in Disaster Communication*, pages 497–518. Springer International Publishing, Cham, 2018. ISBN 978-3-319-63254-4. doi: 10.1007/978-3-319-63254-4_24. URL https://doi.org/10.1007/978-3-319-63254-4_24.

L. Palen, S. Vieweg, S. B. Liu, and A. L. Hughes. Crisis in a networked world: Features of computer-mediated communication in the april 16, 2007, virginia tech event. *Social Science Computer Review*, 27:467, 2009.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10):1345–1359, Oct. 2010. ISSN 1041-4347. doi: 10.1109/TKDE.2009.191. URL http://dx.doi.org/10.1109/TKDE.2009.191.

S. J. Pan, X. Ni, J. Sun, Q. Yang, and Z. Chen. Cross-domain sentiment classification via spectral feature alignment. In M. Rappa, P. Jones, J. Freire, and S. Chakrabarti,

editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 751–760. ACM, 2010. doi: 10.1145/1772690.1772767. URL https://doi.org/10.1145/1772690.1772767.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In M. A. Walker, H. Ji, and A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-1202. URL https://doi.org/10.18653/v1/n18-1202.

B. Plank, A. Johannsen, and A. Søgaard. Importance weighting and unsupervised domain adaptation of POS taggers: a negative result. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 968–973. ACL, 2014. doi: 10.3115/v1/d14-1104. URL https://doi.org/10.3115/v1/d14-1104.

L. Plotnick, S. R. Hiltz, J. A. Kushma, and A. H. Tapia. Red tape: Attitudes and issues related to use of social media by U.S. county-level emergency managers. In L. Palen, M. Büscher, T. Comes, and A. L. Hughes, editors, *12th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Krystiansand, Norway, May 24-27, 2015*. ISCRAM Association, 2015. ISBN 978-8-271-17788-1. URL http://idl.iscram.org/files/lindaplotnick/2015/1225_LindaPlotnick_etal2015.pdf.

H. Purohit, C. Castillo, F. Diaz, A. P. Sheth, and P. Meier. Emergency-relief coordination

on social media: Automatically matching resource requests and offers. *First Monday*, 19 (1), 2014. URL http://firstmonday.org/ojs/index.php/fm/article/view/4848.

J. Qadir, A. Ali, R. U. Rasool, A. Zwitter, A. Sathiaseelan, and J. Crowcroft. Crisis analytics: Big data driven crisis response. *CoRR*, abs/1602.07813, 2016. URL http://arxiv.org/abs/1602.07813.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding with unsupervised learning. *Technical report, OpenAI*, 2018.

R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 759–766, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273592. URL https://doi.org/10.1145/1273496.1273592.

C. Reuter and M.-A. Kaufhold. Fifteen years of social media in emergencies: A retrospective review and future directions for crisis informatics. *Journal of Contingencies and Crisis Management*, 26(1):41–57, 2018. doi: 10.1111/1468-5973.12196. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1468-5973.12196.

C. Reuter, A. Marx, and V. Pipek. Crisis management 2.0: Towards a systematization of social software use in crisis situations. *International Journal of Information Systems for Crisis Response and Management (IJISCRAM)*, 4:1–16, 01 2012.

C. Reuter, T. Ludwig, T. Friberg, S. Pratzler-Wanczura, and A. Gizikis. Social media and emergency services?: Interview study on current and potential use in 7 european countries. *IJISCRAM*, 7(2):36–58, 2015. doi: 10.4018/IJISCRAM.2015040103. URL http://dx.doi.org/10.4018/IJISCRAM.2015040103.

C. Reuter, A. L. Hughes, and M.-A. Kaufhold. Social media in crisis management: An evaluation and analysis of crisis informatics research. *International Journal of Hu-*

man–Computer Interaction, 34(4):280–294, 2018. doi: 10.1080/10447318.2018.1427832. URL https://doi.org/10.1080/10447318.2018.1427832.

S. Ruder. *Neural Transfer Learning for Natural Language Processing*. PhD thesis, National University of Ireland, Galway, 2019.

S. Ruder, P. Ghaffari, and J. G. Breslin. Knowledge adaptation: Teaching to adapt. *CoRR*, abs/1702.02052, 2017a. URL http://arxiv.org/abs/1702.02052.

S. Ruder, P. Ghaffari, and J. G. Breslin. Data selection strategies for multi-domain sentiment analysis. *CoRR*, abs/1702.02426, 2017b. URL http://arxiv.org/abs/1702.02426.

T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 851–860. ACM, 2010. doi: 10.1145/1772690.1772777. URL https://doi.org/10.1145/1772690.1772777.

T. Schnabel, I. Labutov, D. Mimno, and T. Joachims. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 298–307, 2015.

A. Schulz, C. Guckelsberger, and F. Janssen. Semantic abstraction for generalization of tweet classification: An evaluation of incident-related tweets. *Semantic Web*, 8(3):353–372, 2017.

A. Søgaard and M. Haulrich. Sentence-level instance-weighting for graph-based and transition-based dependency parsing. In *Proceedings of the 12th International Conference on Parsing Technologies, IWPT 2011, October 5-7, 2011, Dublin City University, Dubin, Ireland*, pages 43–47. The Association for Computational Linguistics, 2011. URL https://www.aclweb.org/anthology/W11-2906/.

O. Sopova. Domain adaptation for classifying disaster-related twitter data. Master's thesis, Kansas State University, 2017.

K. Starbird, L. Palen, A. L. Hughes, and S. Vieweg. Chatter on the red: what hazards threat reveals about the social life of microblogged information. In K. Inkpen, C. Gutwin, and J. C. Tang, editors, *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work, CSCW 2010, Savannah, Georgia, USA, February 6-10, 2010*, pages 241–250. ACM, 2010. doi: 10.1145/1718918.1718965. URL https://doi.org/10.1145/1718918.1718965.

S. Stefan, B. Deborah, M. Milad, and E. Christian. Sense-making in social media during extreme events. *Journal of Contingencies and Crisis Management*, 26(1):4–15, 2018. doi: 10.1111/1468-5973.12193. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1468-5973.12193.

M. Sugiyama, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1433–1440. Curran Associates, Inc., 2007. URL http://papers.nips.cc/paper/3248-direct-importance-estimation-with-model-selection-and-its-application-to-covaria

B. Sun and K. Saenko. Deep CORAL: correlation alignment for deep domain adaptation. *CoRR*, abs/1607.01719, 2016. URL http://arxiv.org/abs/1607.01719.

B. Sun, J. Feng, and K. Saenko. Return of Frustratingly Easy Domain Adaptation. *ArXiv e-prints*, 2015.

S. Tan, X. Cheng, Y. Wang, and H. Xu. Adapting naive bayes to domain adaptation for sentiment analysis. In M. Boughanem, C. Berrut, J. Mothe, and C. Soulé-Dupuy, editors, *Advances in Information Retrieval, 31th European Conference on IR Research, ECIR 2009, Toulouse, France, April 6-9, 2009. Proceedings*, volume 5478 of *Lecture Notes in Computer Science*, pages 337–349. Springer, 2009. doi: 10.1007/978-3-642-00958-7_31. URL https://doi.org/10.1007/978-3-642-00958-7_31.

A. H. Tapia, K. Bajpai, B. J. Jansen, J. Yen, and L. Giles. Seeking the trustworthy tweet: Can microblogged data fit the information needs of disaster response and humanitarian relief organizations. In *8th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Lisbon, Portugal, 2011.*, pages 1–10, 2011.

A. H. Tapia, K. A. Moore, and N. J. Johnson. Beyond the trustworthy tweet: A deeper understanding of microblogged data use by disaster response and humanitarian relief organizations. In *10th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Baden-Baden, 2013.*, pages 770–778, 2013.

T. Terpstra, A. de Vries, and G. P. R. Stronkman. Towards a realtime twitter analysis during crises for operational crisis management. In *Proceedings of the 9th International Conference on Information Systems for Crisis Response and Management (ISCRAM 2012), Vancouver, Canada*, pages 1—-9, 2012.

Transportation Research Board and National Academies of Sciences, Engineering, and Medicine. *Uses of Social Media to Inform Operational Response and Recovery During an Airport Emergency.* The National Academies Press, Washington, DC, 2017. doi: 10.17226/24871. URL https://www.nap.edu/catalog/24871/uses-of-social-media-to-inform-operational-response-and-recovery-during-an-airport-em

Y. Tsuboi, H. Kashima, S. Hido, S. Bickel, and M. Sugiyama. Direct density ratio estimation for large-scale covariate shift adaptation. *J. Inf. Process.*, 17:138–155, 2009. doi: 10.2197/ipsjjip.17.138. URL https://doi.org/10.2197/ipsjjip.17.138.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.

S. Verma, S. Vieweg, W. J. Corvey, L. Palen, J. H. Martin, M. Palmer, A. Schram, and K. M. Anderson. Natural language processing to the rescue? extracting "situational awareness" tweets during mass emergency. In L. A. Adamic, R. Baeza-Yates, and S. Counts, editors, *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*. The AAAI Press, 2011. URL http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/2834.

S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen. Microblogging during two natural hazards events: what twitter may contribute to situational awareness. In E. D. Mynatt, D. Schoner, G. Fitzpatrick, S. E. Hudson, W. K. Edwards, and T. Rodden, editors, *Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, Atlanta, Georgia, USA, April 10-15, 2010*, pages 1079–1088. ACM, 2010. doi: 10.1145/1753326.1753486. URL https://doi.org/10.1145/1753326.1753486.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390294. URL http://doi.acm.org/10.1145/1390156.1390294.

P. C. Viswa Mani Kiran Peddinti. Domain adaptation in sentiment analysis of twitter. In *Proceedings of the 5th AAAI Conference on Analyzing Microtext*, pages 44—-49, 2011.

P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, and H. Hao. Semantic clustering and convolutional neural network for short text categorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 352–357. The Association for Computer Linguistics, 2015. URL http://aclweb.org/anthology/P/P15/P15-2058.pdf.

H. Watson, R. L. Finn, and K. Wadhwa. Organizational and societal impacts of big data

in crisis management. *Journal of Contingencies and Crisis Management*, 25(1):15—-22, 2017.

Q. Xie, M. Luong, E. H. Hovy, and Q. V. Le. Self-training with noisy student improves imagenet classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10684–10695. IEEE, 2020. doi: 10.1109/CVPR42600.2020.01070. URL https://doi.org/10.1109/CVPR42600.2020.01070.

X. Yang, C. MacDonald, and I. Ounis. Using word embeddings in twitter election classification. *CoRR*, abs/1606.07006, 2016. URL http://arxiv.org/abs/1606.07006.

F. Yao and Y. Wang. Domain-specific sentiment analysis for tweets during hurricanes (DSSA-H): A domain-adversarial neural-network-based approach. *Comput. Environ. Urban Syst.*, 83:101522, 2020. doi: 10.1016/j.compenvurbsys.2020.101522. URL https://doi.org/10.1016/j.compenvurbsys.2020.101522.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, pages 189–196, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics. doi: 10.3115/981658.981684. URL http://dx.doi.org/10.3115/981658.981684.

Z. Ye, Y. Geng, J. Chen, J. Chen, X. Xu, S. Zheng, F. Wang, J. Zhang, and H. Chen. Zero-shot text classification via reinforced self-training. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3014–3024. Association for Computational Linguistics, 2020. URL https://www.aclweb.org/anthology/2020.acl-main.272/.

J. Yin, A. Lampert, M. A. Cameron, B. Robinson, and R. Power. Using social media to

enhance emergency situation awareness. *IEEE Intelligent Systems*, 27(6):52–59, 2012. doi: 10.1109/MIS.2012.6. URL https://doi.org/10.1109/MIS.2012.6.

Z. Yin, V. Sachidananda, and B. Prabhakar. The global anchor method for quantifying linguistic shifts and domain adaptation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9412–9423. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8152-the-global-anchor-method-for-quantifying-linguistic-shifts-and-domain-adaptation.pdf.

K. Zhang, M. Gong, and B. Schölkopf. Multi-source domain adaptation: A causal view. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3150–3157. AAAI Press, 2015. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/10052/9994.

S. Zhang and S. Vucetic. Semi-supervised discovery of informative tweets during the emerging disasters. *CoRR*, abs/1610.03750, 2016. URL http://arxiv.org/abs/1610.03750.

H. Zhao, S. Zhang, G. Wu, J. P. Costeira, J. M. F. Moura, and G. J. Gordon. Multiple source domain adaptation with adversarial training of neural networks. *CoRR*, abs/1705.09684, 2017. URL http://arxiv.org/abs/1705.09684.

G. Zhou, Z. Xie, J. X. Huang, and T. He. Bi-transferring deep neural networks for domain adaptation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. doi: 10.18653/v1/p16-1031. URL https://doi.org/10.18653/v1/p16-1031.

F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He. Supervised representation learning: Transfer learning with deep autoencoders. In Q. Yang and M. J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence,*

IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, pages 4119–4125. AAAI Press, 2015. URL http://ijcai.org/Abstract/15/578.

Y. Ziser and R. Reichart. Neural structural correspondence learning for domain adaptation. In R. Levy and L. Specia, editors, *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), Vancouver, Canada, August 3-4, 2017*, pages 400–410. Association for Computational Linguistics, 2017. doi: 10.18653/v1/K17-1040. URL https://doi.org/10.18653/v1/K17-1040.

Y. Ziser and R. Reichart. Pivot based language modeling for improved neural domain adaptation. In M. A. Walker, H. Ji, and A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1241–1251. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-1112. URL https://doi.org/10.18653/v1/n18-1112.