Disaster Tweet Classification using Parts-of-Speech Tags:

A Domain Adaptation Approach

by

Tyler Robinson

B.S., Kansas State University, 2014

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTERS OF SCIENCE

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2016

Approved by:

Major Professor
Professor Doina Caragea

Copyright

Abstract

Twitter is one of the most active social media sites today. Almost everyone is using it, as it is a medium by which people stay in touch and inform others about events in their lives. Among many other types of events, people tweet about disaster events. Both man made and natural disasters, unfortunately, occur all the time. When these tragedies transpire, people tend to cope in their own ways. One of the most popular ways people convey their feelings towards disaster events is by offering or asking for support, providing valuable information about the disaster, and voicing their disapproval towards those who may be the cause. However, not all of the tweets posted during a disaster are guaranteed to be useful or informative to authorities nor to the general public. As the number of tweets that are posted during a disaster can reach the hundred thousands range, it is necessary to automatically distinguish tweets that provide useful information from those that don't. Manual annotation cannot scale up to the large number of tweets, as it takes significant time and effort, which makes it unsuitable for real-time disaster tweet annotation. Alternatively, supervised machine learning has been traditionally used to learn classifiers that can quickly annotate new unseen tweets. But supervised machine learning algorithms make use of labeled training data from the disaster of interest, which is presumably not available for a current target disaster. However, it is reasonable to assume that some amount of labeled data is available for a prior source disaster. Therefore, domain adaptation algorithms that make use of labeled data from a source disaster to learn classifiers for the target disaster provide a promising direction in the area of tweet classification for disaster management. In prior work, domain adaptation algorithms have been trained based on tweets represented as bag-of-words. In this research, I studied the effect of Part of Speech (POS) tag unigrams and

bigrams on the performance of the domain adaptation classifiers. Specifically, I used POS tag unigram and bigram features in conjunction with a Naive Bayes Domain Adaptation algorithm to learn classifiers from source labeled data together with target unlabeled data, and subsequently used the resulting classifiers to classify target disaster tweets. The main research question addressed through this work was if the POS tags can help improve the performance of the classifiers learned from tweet bag-of-words representations only. Experimental results have shown that the POS tags can improve the performance of the classifiers learned from words only, but not always. Furthermore, the results of the experiments show that POS tag bigrams contain more information as compared to POS tag unigrams, as the classifiers learned from bigrams have better performance than those learned from unigrams.

Table of Contents

List of Figures

Chapter 1

Introduction

Twitter is one of the most active social media sites today. Everyone is using it, from world famous musicians and politicians to the average Joe. It is a medium by which people stay in touch and inform others about events in their lives. One such event is disasters. Both man made and natural disasters unfortunately occur all the time. When these tragedies transpire, people tend to cope in their own ways. One of the most popular ways people convey their feelings towards these events is by offering or asking for support, providing valuable information about the disaster, and voicing their disapproval towards those who may be the cause. However, not all of the tweets are guaranteed to be useful or informative. The number of tweets that are posted during these disasters increases upwards to a number in the hundred thousands range, and some will be useful and provide important information and others will not.

1.1 Motivation

The unfortunate truth about disaster tweets is that because of the massive number of tweets that are tweeted every day it is very easy for useful information to become lost within the

tides of text. To be able to utilize that information it is needed for someone to go through all of those tweets and annotate and categorize them to better determine what information is important or not. This process can take a significant amount of time, that we may not have, to complete. And even if we do manage to get relevant information from the data in a timely manner it is only useful for the one disaster and if another disaster were to arise the whole process would have to be redone. To be able to determine whether a tweet is useful or not, meaning that it contains important information that could be used to help recover from the disaster, in a timely manner, we must find a way to quickly and accurately extract and classify them as such. This information contained by disaster tweets can then be used to help both authorities as well as the general public deal with a specific disaster.

1.2 Problem Addressed

When a disaster occurs, if we want to classify that information using text classification, we need pre-labeled data that can be used to generate a classifier. As stated above the process of getting that pre-labeled data can take much time to generate. However, it is reasonable to assume that pre-labeled data from previous disaster is available. Another major issue is that it is difficult to use a classifier generated for one disaster on another disaster, because the attributes or words used in the case of tweets can differ vastly between a disaster about a hurricane and a disaster about a shooting. Therefore, the problem that we address is how to learn from a source disaster to a target disaster in a Domain Adaptation scenario where unlabeled data from target can be used together with labeled data from the source.

1.3 High Level Approach

In previous work in the MLB group at Kansas State University, we have been using a domain adaptation algorithm to generate a classifier model to classify new information. We took all the tweets of the source data and converted them into binary word vectors where each word is considered a feature. With this approach we have received limited success. In my experiments I will be creating part of speech (POS) tag features to be used both by themselves and to be used in addition to the bag-of-words representation from previous experiments when generating a text classifier. The goal of my experiment is to study the effect of POS tags on the performance of the generated text classifier.

Chapter 2

Background

2.1 Text Classification

Text classification is the process of labeling different texts with some labels, such as *on-topic/off-topic*, or *provides/does not provide supporting information*. With tweets, the process of labeling can be difficult. Tweets do not tend to have a full linguistic structure. They are not always a straightforward sentence that is easy to understand. Some can have a couple of words and others can be a jumble of random characters (emojis). In the context of disaster management, tweets can be labeled as either on-topic or off-topic in regards to a given disaster. In relation to the Sandy Hurricane that occurred in 2012, an example of an on-topic tweet would look something like: *12 ft storm surge expected.....hurricane Irene was 4.5 feet haha yea keep taking this hurricane lightly* and an off-topic tweet would look something like: *Everyone and their mothers are at home depot.* There several approaches to labeling tweets. One approach that has been used many times is to use experts in a field of interest, to manually go through and annotate data that revolve around the information you are wanting to label. This is a long and time consuming process that may not produce enough information quickly to be usable. Furthermore, the people analyzing the data may

not classify it as accurately as they would hope. To improve on this process machine learning (ML) can be used, as described in the next section.

2.2 Machine Learning

ML is a commonly used paradigm that attempts to either determine unknown information about data from known information in the context of supervised learning or identify structure in data (unsupervised learning).

2.2.1 Supervised Learning

Supervised learning is where we use training data that has some desired label, to learn classifiers that can help classify new data[McCallum et al., 1998]. By learning from the characteristics and the categories of interest of preclassified documents [Sebastiani, 2002] or pre-labeled data, ML tries to classify new data. One common use for ML is to facilitate text classification, which we can perform by constructing a program that builds a classifier based on existing labeled information. With that new classifier, the program would then analyze and label new data according to predetermined categories, such as whether the text is providing useful information or not. When trying to perform any kind of text classification the data used is not usually a pure sentence. The data used in ML is normally represented as a N-dimensional feature vector $x = (x_1, ..., x_N) \in \mathbf{R}^N$, where each dimension is called a feature [Zhu and Goldberg, 2009]. When it comes to data classification, there are many algorithms available for use. For the purposes of this paper, I will only go briefly into the specific one I used, Naive Bayesian classification.

Naive Bayes is an algorithm for predicting the class of a set of unknown data,in particular, documents in the context of text classification [McCallum et al., 1998]. It is often used

for very large sets of data, it uses words as features and it assumes that every feature is independent of all others. When learning a Naive Bayes classifier you need to estimate the prior and likelihood. The Prior is the probability of a classification and likelihood is the probability of an feature given the classification. Based on these, we can find the posterior probability of a label given an instance by multiplying the prior and likelihood together and dividing that by the probability of the instance.

$$\begin{aligned} Prior &=> P(C) \\ Likelihood &=> P(X|C) \\ Posterior &=> P(C|X) = \frac{P(X|C)P(C)}{P(X)} \end{aligned} \tag{2.1}$$

Where X is the set of features $x_1, x_2, \ldots, x_n$ and C is the set of classifications $c_1, c_2, \ldots, c_m$. Under the assumption that words are independent, the above algorithm can be further simplified into:

$$P(C|X) \propto \prod_{t=1}^{N} P(x_t|C) \tag{2.2}$$

x_t indicates whether the word t occurs in the instance x or not.

The Naive Bayes algorithm described above is also known as Bernoulli Naive Bayes, as it uses binary feature word vectors to represent documents, where a 0 means that the word does not appear in that document and a 1 means that it does[McCallum et al., 1998]. The Multinomial Naive Bayes variant uses a word count feature vector where each count represents the number of times that word appears in that document. Both methods of generating classifiers are widely used for specific scenarios. However, we used Bernoulli Naive Bayes here given that tweets are short and don't have many representations of the same word.

2.2.2 Semi-Supervised Learning

There is a hybrid approach to ML called semi-supervised learning,[Zhu, 2005] where we have a small amount of labeled data and a large amount of unlabeled that is used to construct the classifier. However, given the small amount of labeled data, semi-supervised classfiers can easily drift away from the correct classification boundary. Furthermore, when a disaster is just happening, we cant assume that we have any labeled data from that disaster itself but it is reasonable to assume that we have labeled data from a prior disaster. Therefore, in this work, I will be turning my focus towards a technique called domain adaptation, where I attempt to construct a classifier based on labeled data from a source disaster and unlabeled data from a target disaster.

2.2.3 Domain Adaptation

Domain Adaptation (DA) is a process of constructing a classifier based on some labeled source domain together with unlabeled data from the target domain, and using that newly created classifier to label documents in the target domain. A domain is simply a set of data that share common attributes. In this work, the different domains will be different sets of tweets from multiple disasters around the world. In relation to DA the source that I will be training my classifier off of will be labeled tweets from past disasters and the target unlabeled data would be from current disasters. One problem with DA is determining what types of attributes will transfer between two domains. Taking a look at tweets generated during Hurricane Sandy, we see a distinct set of popular words used to help provide important information. If we use those words as attributes, we can construct a classifier that would be successful in extracting important information from other sets of tweets about hurricanes. If we take that same set and try to classify information on a different domain, such as the Boston Bombing, the classifier would have a more difficult time being as accurate. Although

both sets of information are similar in the sense that they are both tweets that contain information about disasters, there can be more differences than similarities. The reason for this is due to the fact that some terminology will be similar between the two different domains, but much will not. The Sandy Hurricane is an uncontrollable disaster, while the Boston Bombing is a man-made disaster that no one could have predicted. The hurricane tweets would most likely contain information about where people could go to find shelter and be calling out to others for help. The bombing tweets would more likely be focused on how the event happened and who was involved and why. Domain Adaptation focuses on the similarities between the disasters and construct a classifier that could be used to label information from a new target disaster based on the labeled data from a prior disaster and unlabeled data from the target itself.

2.3 Part of Speech Tagging

POS tags are another way of labeling text data, in particular we are labeling the tokens/words in a text. Examples of POS tags are nouns, verbs, and adjectives. In a way POS tags are in themselves features of words; they give classification and context to words that allow us to better understand the purpose of word choice and the meaning of sentences as a whole. Figure 2.1 shows an example of a full sentence and the POS tags that have been assigned to them.

Sentence	A	disaster	tweet	can	contain	useful	information!
POS Tag	Determiner	Noun	Noun	Verb	Verb	Adjective	Noun

Figure 2.1: An example of a full sentence and the part of speech tags associated with each word.

I wanted to see if, linguistically, both relevant and irrelevant tweets had some similar trend when it came to the type of POS being used. Tagging text provides a great deal

of information beyond simply an unanalyzed sequence of words. Although less informative than a full syntactic parse that additionally provides higher-level structural information and relations between words, POS tagging can be done much more accurately and quickly than parsing [Dale, 2000]. Similar to ML, POS tagging can also be broken down into the two subfields of supervised and unsupervised. Both are similar to ML's counterparts, supervised POS tagging works off of a predefined library of tagged text and unsupervised uses complex algorithms to determine the POS tags based on sentence structures and attributes. The only problem with attempting to tag tweets is that most POS taggers are written for full, complete sentences. Unfortunately tweets are not always well structured or coherent. A solution I used to counter this issue was by using a specially designed tweet parser and annotator tool suite, developed by a research group at Carnegie Mellon University, called ARK-NLP [Owoputi et al., 2013]. This allowed me to assign POS tags to strings of tweets and characters such as emojis and abbreviations that normal natural language processors would have trouble parsing. A normal sentence would have a concise structure, for example the sentence *The quick brown fox jumped over the lazy dog* has a subject acting upon another object. If we look at the tweet *kay smh to wat just happened!*, it has a more conversational tone has abbreviations and slang terms that many normal POS taggers would miss. The process of tagging tweets is done by marking items that are specific to tweets with their own special tag [Gimpel et al., 2011]. Some examples of these special tags are # for hashtags and @ for at-mentions. The research group at Carnegie Mellon University, with the assistance of some pre-established English POS taggers went through a series of stages to accurately annotate tweets with their POS tags for the use as a tagset for their parser. After the initial tagging the researchers manually analyzed the results, removed any non-english words found and constructed updated tagging guidelines to improve the tokenizers used in the parsing process, and constructed a tagset of words/symbols with a defined POS tag. The parser had two different sets of classifications for POS tags. The individual POS tag, such as nouns,

verbs, and adjectives, with some tweet specific tags like hashtags and at-mentions, and the POS groups such as nominal, closed-class, and Twitter/online-specific [Gimpel et al., 2011]. Examples of these groupings are nominal tags where words are common nouns (books, someone), pronouns (it, you, u, meeee), proper nouns + possessive (America's), etc. and closed-class words are like determiners (the, teh, its, it's), or coordinating conjunctions (and, n, &, +, BUT) [Gimpel et al., 2011]. The individual tags and not their category groupings were used because I wanted to see what individual POS tags would have to say about a tweet when trying to classify it.

Chapter 3

Approach

3.1 Domain Adaptation Approach

Figure 3.1 shows the Naive Bayes DA algorithm and explains how it was used. In this algorithm I denote the target test data with TT the train target unlabeled data by tTU, the training source labeled data by tSL and test unlabeled validation data with tUV. The next subsections describe in macro what my three individual sets of features would be and I will go into further detail in a later section. All three sets of features will use this Domain Adaptation algorithm for conducting the experiments.

3.1.1 Bag of Words

In the past experiments done by Li et al. [2015] they converted the data into a bag of words representation. Due to some difference in the cleaned data, which I will talk about more later, I decided as a base test that I would repeat the same bag of words test with my set of data. For this, I took each tweet and with the help of the program Weka, converted each tweet into word vectors. We used words as attributes of a tweet, and used a binary representation, ie. each tweet instance either had a word (1) or did not (0). I then used

this newly generated data as my tSL, tTU and TT for my experiment. The sole goal of this experiment was to create the base case of what has already been done, so that I could compare the results of my later approaches to see if there was an improvement.

3.1.2 Bag of POS Tags

For this set of experiments, before using Weka to convert it into its final form for testing I needed to extract the POS tags for each tweet and replace the word with its corresponding POS tag. Once I had the POS tag representation of each tweet I then did the same process as above, I converted each tweet in POS form into a word vector. Each POS tag was then considered its own attribute of a tweet. In this approach I wanted to see if how much information the POS tags carry by themselves with respect to the tweet.

3.1.3 Bag of Words + Bag of POS Tags

This approach was to see if combining the bag of words and bag of POS tags approaches would yield better results than using either one by itself. To conduct this experiment all I did was take the cleaned tweets and append their POS tags, so that each instance would have both sets of attributes when constructing a classifier.

3.1.4 Bag of POS Tag Bi-grams

This approach delved further into POS tags. I took the POS tags extracted during the approach stated in subsection 3.1.2. I looped through each set of POS tags and constructed bi-grams. Once I had all of the bi-grams for data set I converted each tweet in POS bi-gram form into a word vector each b-gram was then considered its own attribute of a tweet. In this approach I wanted to see if the bi-grams of POS tags could reveal more information than the POS tags alone.

3.1.5 Bag of Words + Bag of POS Tag Bi-grams

For this final test I wanted to see if the classifier of combining the POS tag bi-grams with the text would yield better results to that of either feature set by themselves as well as when compared to the classifier produced by the approach in subsection 3.1.3. To conduct this experiment I took the sets of features from subsections 3.1.1 and 3.1.4.

1. First we calculate the initial Naive Bayes Classifier by calculating the likelihood and prior of the training Source Labeled (tSL) data.
The likelihood is $P_{tSL}(X|C)$ where X is a feature and C is the classifier (on-topic/off-topic).
The prior is $P_{tSL}(C)$.

2. Based on the generated classifier learned from the source data, we assign new labels to tTU. We then take the probabilities of each instance in tTU for the on-topic and off-topic classifications, and order them from greatest to least. Based on the ratio of on-topic to off-topic in the target documents, we then take that many hard-labeled instances on the tTU, and add them to the training labeled data.

3. With the new labels in tTU, we loop and calculate the updated classifier based on both tSL and labeled tTU until the labels of the remaining instances in tTU stop changing, as described below.

Maximization Step: We first calculate the prior and likelihood of both tSL and tTU. The priors being $P_{tSL}(C)$ and $P_{tTU}(C)$. We then calculate the likelihoods, which are $P_{tSL}(X|C)$ and $P_{tTU}(X|C)$. Once those are calculated to get the overall prior and likelihood, we add both probabilities together with tSL starting off with higher weight than tTU due to the fact that tSL contains more labeled data. As the loop iterates the weight slowly shifts from tSL to tTU as there will be more labeled data in tTU.
Specifically,

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

where $\gamma = min(\tau\delta, 1)$, where τ is the loop iteration indicator, δ is the value that determines the speed at which we shift the weight from tSL to tTU.

Expectation Step: We calculate the class distribution for the remaining part of the tTU data.

4. Once the classifier generating loop has finished, we take that final classifier and use it on the TT data to label the rest of the information.

Figure 3.1: Domain Adaptation algorithm

Chapter 4

Experimental Setup

In this chapter, I will discuss the questions that I am trying to answer with my experiment, as well as the data and the pre-processing I did to prepare the data for my experiments.

4.1 Research Questions

There are several main questions that I wanted to answer by conducting my experiments. These questing are:

1. In terms of classification, how accurate is a classifier that is generated from only POS tags when labeling new data? How accurate is a classifier generated from the bi-grams of POS tags?

2. Does knowing the part of speech of a tweet reveal new information, previously unknown with just the text?

3. Does the combination of text + POS tags result in improved classifications as compared to those labeled from either text or POS tags alone? Is this combination better than the combination of text + bi-grams of POS tags?

The experiments are designed to answer these questions.

4.2 Data Used

The data used in this work is a collection of tweets posted during different disasters. This data was published by Olteanu et al. [2014] as an on-line repository of different disaster tweets, for which labels have been assigned. For my experiment I had six different sets of disaster tweets starting in 2012 and ending in 2013. Each dataset roughly contained about 10,000 tweets each, with each instance being pre-labeled with on-topic or off-topic. Figure 4.1 shows the details of the data distribution of on-topic/off-topic for before and after the text preprocessing.

Disaster	On-Topic	Off-Topic	Total	On-Topic	Off-Topic	Total
Sandy Hurricane	6,138	3,870	10,008	5,257	3,741	8,998
Alberta Flood	5,189	4,841	10,030	3,491	4,703	8,194
Boston Bombing	5,648	4,364	10,012	4,433	4,301	8,734
Oklahoma Tornado	4,827	5,165	9,992	3,204	5,044	8,248
Queensland Flood	5,414	4,619	10,033	3,232	4,539	7,771
West-Texas Explosion	5,246	4,760	10,006	4,119	4,726	8,845
		Raw			**Clean**	

Figure 4.1: This table represents the number of tweets that I started with that were labeled as on-topic and off-topic as well as the number of tweets remaining after I parsed and cleaned each tweet.

For the five different approaches described in section 3.1, I created seven different source/target pairs of disaster data. Figure 4.2 shows these different pairs. For pairs one through five I used the 2012 Sandy Hurricane as my source disaster data with a different type of disaster as the unlabeled target data. Pairs six and seven were then disasters that were of the same type of disaster for both source and target.

Before I could run my experiments I had to first parse and extract POS tags from each tweet then clean each text body. The data given to me was in the form of a csv file, which

Pair	Source Disaster Data	Target Disaster Data
P1	2012 Sandy Hurricane	2013 Queensland Floods
P2	2012 Sandy Hurricane	2013 Boston Bombing
P3	2012 Sandy Hurricane	2013 West Texas Explosion
P4	2012 Sandy Hurricane	2013 Oklahoma Tornado
P5	2012 Sandy Hurricane	2013 Alberta Floods
P6	2013 Boston Bombings	2013 West Texas Explosion
P7	2013 Queensland Floods	2013 Alberta Floods

Figure 4.2: Pairs of Source/Target disasters used for the experiments. Each row represents its own separate experiment. For each pair I ran three tests: Text only, POS tags Only, and Text + POS tags.

I used to conduct my preprocessing.

4.3 Preprocessing

There were two main tasks I needed to perform before I could conduct my experiments, the first being I needed to extract the POS tags from the original, unmodified tweet text, and second, clean and modify the tweet text to remove irrelevant data.

4.3.1 POS Tag Extraction

To extract the POS tags from each tweet I needed to utilize the Ark-NLP tweet parser and tagger Twokenizer. Twokenizer was used to both parse and tag each tweet based on the tweet's linguistic structure, which Twokenizer would attempt to determine. For my experiments, I was given a comma separated file with every tweet, ID, and their predetermined label. I extracted from each line, the tweet string and passed the data to the Twokenizer. The set of information produced by the Twokenizer was the original tweet string, the POS tags for each word/symbol in the tweet string, and the Twokenizer's confidence score about how accurate it believes each tweet to be. An example of the output of the Twokenizer

12	$	0.9926
ft	V	0.5285
storm	N	0.7717
surge	N	0.9455
expected	V	0.9741
.....	,	0.9901
hurricane	N	0.6422
Irene	^	0.9499
was	V	0.9924
4.5	$	0.9093
feet	N	0.9923
haha	!	0.9945
yea	!	0.9931
keep	V	0.9366
taking	V	0.9987
this	D	0.9613
hurricane	N	0.9761
lightly	R	0.9804

(a) This is an example of the information produced by the Twokenizer, formatted in an easier to read format. The left column represents the words/symbols used in a tweet. The center represents the tagged Part Of Speech for the corresponding word/symbol. The right column represents the tagger's confidence from 0-1 that it was in correctly assigning the appropriate tag for the word/symbol.

N	common noun
O	pronoun
S	nominal-possessive
^	proper noun
Z	proper noun + possessive
L	nominal + verbal
M	proper noun + verbal
V	verb
A	adjective
R	adverb
!	interjection
D	determiner
P	pre- or Postposition
&	coordinating conjunction
T	verb particle
X	existential there
Y	X + Verbal
#	hashtag
@	at-mention
~	discourse marker
U	URL or Email
E	emoticon
$	numeral
,	punctuation
G	other abbreviations, foreign words

(b) This is a table pulled from [Gimpel et al., 2011], and it provides a useful reference guide to what the tags mean.

given the tweet *12 ft storm surge expected ..... hurricane Irene was 4.5 feet haha yea keep taking this hurricane lightly*, is shown in Figure 4.3a.

The POS tags are then assigned to their corresponding words where I then constructed a new Weka arff file using Weka's open source libraries that had the attributes of Text, POS Tags, and the tweet's labeled classification. From here, I proceeded with the cleanup of the original text to remove the unwanted information.

4.3.2 Bi-gram Construction

To construct the bi-grams of the POS tags I took the results from the previous section and looped through them. I recorded every possible bi-gram of the POS tag attributes, and using a sliding window looped through the string that contained the POS tags. This window would show only two tags at any time and would shift over one tag on every loop iteration so that every tag, except for the first and last tag, could have a chance to be first tag and the second tag in the bi-gram. I took these bi-grams and proceeded to construct a new Weka arff file where the attributes were the Text, POS tag bi-grams, and the tweet's labeled classification.

4.3.3 Text Preprocessing

To improve the reliability of the generated classifier, I needed to clean the text of unnecessary noise. I followed the same process as [Li et al., 2015] for my cleaning steps. I removed non-pritable ASCII characters as well as substituting Usernames, URLs, and email addresses with placeholders such as USERNAME/EMAIL/URL. Through my process of extracting POS tags I also removed any duplicate tweets that may have been there due to re-tweets. Figure 4.1 shows the state of the data before and after I cleaned the information. As you can see, the total number of dirty tweets stayed relatively the same and the total number of clean tweets for each disaster also stayed relatively the same.

4.4 Experimental Design

My experiments were run based on the algorithms used by [Li et al., 2015] thus the terminology used is based on their work. I used a 5-fold cross-validation on my Target data to prepare it for my experiment. The target set was evenly split into five parts with equal ratios of on-topic to off-topic classifications. Figure4.4 shows an example of a 5-fold cross

validation where we have some data and we, to the best of our ability, split up the data into 5 random, but equal, parts. We take one fold and set it aside to be used as the validation test data and the other 4 folds are used to construct the classifier. By allowing each subset a chance to be the test data the variance between the results can be a lot less, though this process will take k times to run since it has to run over each subset. For each pair of disaster data, I ran the experiment 5 times. Each time, I selected one fold to be the target Test (TT) data, the next three folds as the test target unlabeled (tTU) data. The final fold was saved for future experiments and not used.

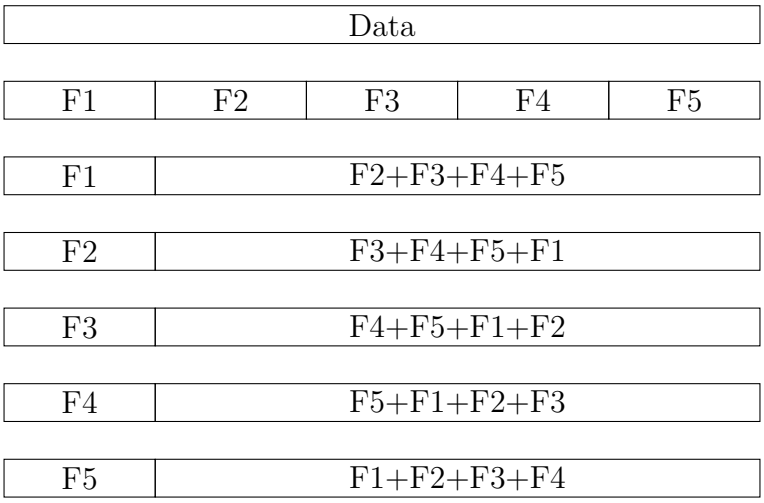

Figure 4.4: A physical representation of how k-fold cross validation would work. In this example the data is split up into five as equal as possible folds.

4.4.1 Parameter Tuning

Due to the number of tweets per disaster set we took 4 random subsets of different sizes of tSL to count as our source to learn from. The numbers of the subsets were: (500, 1000, 2000, 4000).To determine how fast we should shift the weight from the source to the target and how many instances from tTU to take as hard-labeled on-topic and off-topic for self-training, we conducted a validation step. For each iteration of tTU and TT we took one

fold of the 3-fold tTU and used that as the test validation (TV) data. We then took the remaining 2 folds of tTU as the test unlabeled validation (tUV) data. From there we trained our classifier with tUV+tSL and tested on TV. This validation step was conducted multiple times in an attempt to determine the best values stated above. The values we decided to use for how fast to shift the weight from source to target were: (0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9) and the number of on-topic and off-topic instances that we labeled from the target were: (1:1, 5:5, 10:10). After the five rounds of experiments, each fold will have been used as TT and tTU.

The number of combinations of parameters that we run through during the tuning process for each set of experiments can be CPU intensive and can take much time to complete. In perspective for each of the 7 pairs we ran our experiment 5 times for each tSL/tTU combinations. Each tSL/tTU experiment we then need to run that test 5 times for each number of instances to take from the source for the purpose of learning the classifier. Each instance is then run each subset with the 11 different weights on the target data and for each weight we try the 3 different numbers of on-topic/off-topic data to take from the target labeled data. In total for one experiment we have to run our algorithm with the different values 4,620 times. Since I had 5 experiments I ran the algorithm a total of 23,100 times. Doing this amount of computations would take an extremely long amount of time on my personal computer and I would not be able utilize my computer during that time fully. Because of this I conducted all of my experiments on Kansas State University's high performance computing cluster, "Beocat". Beocat allows me to throw more resources at my computations to increase the speed and decrease the time it takes to run.

4.5 Evaluation

With this problem, it can be difficult estimating the performance of a classifier. One way of addressing this is to first remove a portion of the data to be used in construction of the classifier, to be used later as a way of verifying the success of the construction[Kohavi et al., 1995]. This process of removing data, constructing the new classifier and using the removed data to verify the correctness of the classifier is a evaluation model known as Cross Validation. There are many different cross validation models that produce similar results to one another. Two of these models are holdout and K-fold cross validation. The holdout model is probably the simplest. The process done is to randomly split the data into two separate test sets where one will be used for creating the classifier and the other used to test. This is the simplest due to the fact that it seems to produce result relatively quickly though the results can vary when different splits are used. Because, how successful the classifier construction is entirely based on how we split the data, K-fold cross validation is a little more complicated but will produce better performance estimates. This is done by splitting the data into k-subsets and looping through all of the resulting sets, using each one as the test data and the remaining k-1 sets as the classifier trainer.

Chapter 5

Experimental Results and Discussion

Due to my cleaning methods, the number of remaining cleaned instances was slightly different from that of [Li et al., 2015]. I could not directly use the result of their tests on just the bag of words representation of the text only because of this. So, I conducted five experiments in total. The first test was to set a baseline by just running the DA algorithm constructed by [Li et al., 2015] with my version of the cleaned text only. The other four tests were then testing to see if only the POS tags and POS tag bi-grams would result in an accurate classifier and whether a combination of both POS tags and the text or POS tag bi-grams and text would provide a better classifier as compared to using just POS tags, POS tag bi-grams, or text alone.

For the five tests I had a set of seven source/target pairs. For the bulk of the tests, I used 2012 Sandy Hurricane tweets as the Source to see if I could generate a classifier that would work on information that was from two different types of disasters. My main test was to see what kind of results I would get with POS tags only in comparison to the baseline test of text only as well as POS tag bi-grams only in comparison to the baseline test. Once my first two sets of experiments were done, I also ran an experiment to see if text and POS tags together or POS tag bi-grams and text can produce a classifier better then those

obtained from each alone. I followed the first set of pairs in [Li et al., 2015], and tested to see if the difference in time of each disaster's occurrence had any correlation to the success of the generated classifier. I followed this because I felt that this would have best success of producing results to answer my initial questions due to the source and target pairs being in different domains of disaster data. My intuition on the result of my experiment was that I did not believe that solely knowing the POS tags would yield significant improvements. This is due to the fact that because the chances of a on-topic and off-topic tweet containing a common POS tag such as a verb or noun, are high, which means that they would be less helpful in identifying the proper classification. I know that my results for the text only, will likely be slightly less successful then that of [Li et al., 2015] because of the number of test data after my cleaning is smaller. My hypothesis is that with the addition of POS tags to the existing cleaned text, the classifier produced will be more successful than the text alone.

5.1 Results

The output of my test consisted of the percent accuracy of correctly classified target instances and the area under the receiver operator curve(ROC). I have averaged out the results over the five folds and present them in figures 5.1 and 5.2 as accuracy and auROC respectively.

As you can see the five columns in each set of three represent the test for just text, POS, text+POS, POS tag bi-grams, and POS tag bi-grams+text. Text alone seemed to have the most successful results of the five. I was not surprised to see that the POS only tests did significantly less in comparison. I did not think that the knowledge of whether a specific tweet had a certain part of speech tag, would yield high results since every tweet would have a high chance of have nouns or verbs. What I did find surprising is that with the addition of POS tags along with the text, the results were generally worse than with the text alone. This is due to the fact that common POS tags such as verb or nouns appear consistently

in both off-topic and on-topic tweets. This effects how the experiment created the classifier because that means those tags are less useful and only produce noise. The added noise make the classification job more difficult. Looking at the results of the POS tag bi-grams seemed to produce better results in comparison to the POS tags alone, but worse than the text alone.

5.2 Discussion

Although sometimes successful, the average accuracy of labeling of each tweet still went down by roughly 2% between the text tests and the text+POS tests. The POS tag bi-grams tests seemed to have a better accuracy compared to those tests with POS tags but worse than the text alone tests. Same with the average AuROC, although less noticeably. Taking a closer look at the average AuROC, in two cases, there was a slight increase. Looking at Figure 5.2, tests pair with Sandy Hurricane as the source and Queensland Floods as the target and the test pair with Sandy Hurricane as the source and West Texas Explosion, both show this increase. I conducted a paired t-test on the text only experiments with the text+POS tag experiments and text only with text+POS tag bigrams. The p-score of these tests revealed that though the resulting accuracy and AuROC values of text+POS tag and text+POS tag bigram, the difference was not statistically significant which means that the POS tags and bigrams did not harm the source and target data. As a whole though, looking at the data it is clear that the results were hindered by the additional attributes and that our classification would have had a more positive performance without this addition.

With these results I can now answer some of the questions I raised earlier.

1. In terms of classification, how accurate is a classifier that is generated from only POS tags when labeling new data?

   In relation to the base case of text only, the classification was less accurate when POS

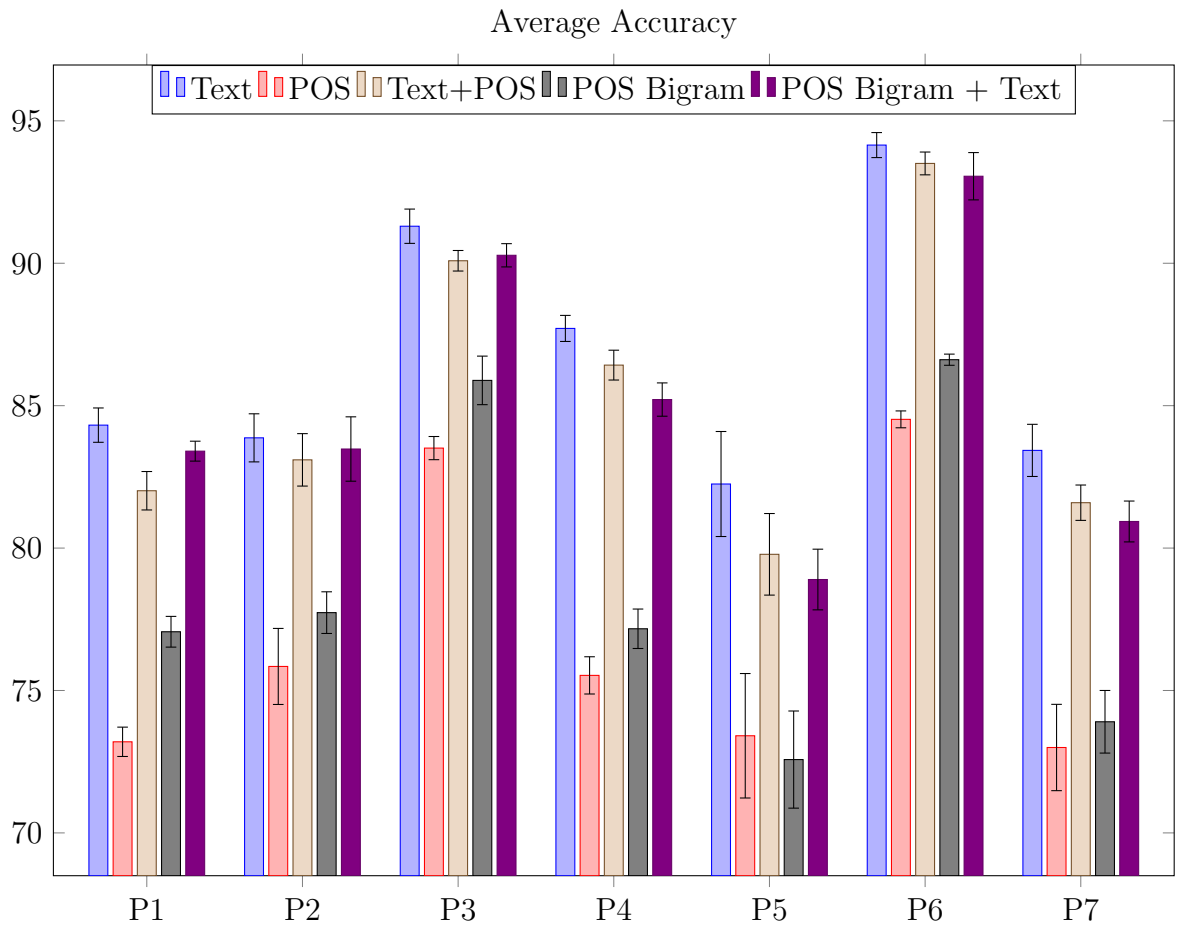

Figure 5.1: This table shows the average accuracy out of 100 for the 5 sets of tests on the seven different disaster pairs. The five bars in each pair represents the average of all five folds in a 5-fold cross validation experiment. The error bar for each experiments shows the standard deviation of the results across the 5 folds. The far left bar of each set represents the base case of the bag of words only. The other bars represent my experiments. Going from left to right the five bars represent Text Bag of Words, POS tags, POS tags + Bag of words, POS tag Bigrams, and POS tag Bigrams + Bag of words. At a glance it can be seen that on average the POS tag experiment was less successful than all the other experiments except for pair 5 which was Sandy Hurricane as the source and Alberta Floods as the target when compared to the POS Bigram experiment. We can also see that the error bars on most of the results are relatively small, thus the results were very close in value between each fold.

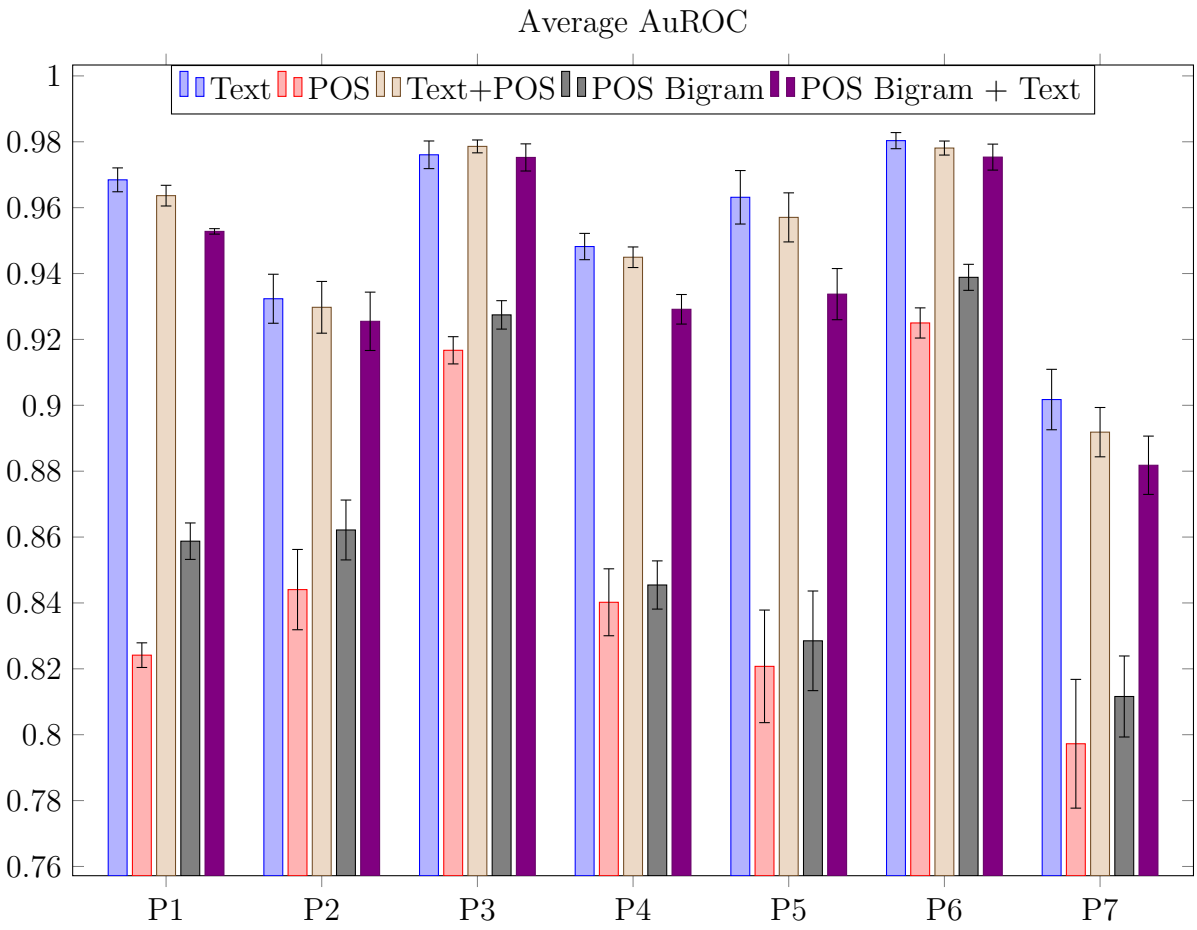

Figure 5.2: This table shows the average AuROC out from 0 to 1 for the 5 sets of tests on the seven different disaster pairs. The five bars in each pair represents the average fo all 5-fold cross validation experiments. The error bar for each experiments shows the standard deviation of the results across the 5 folds. The far left bar of each set represents the base case of the bag of words only. The other bars represent my experiments. Going from left to right the five bars represent Text Bag of Words, POS tags, POS tags + Bag of words, POS tag Bigrams, and POS tag Bigrams + Bag of words. At a glance it can be seen that on average the POS tag experiment was less successful and than all the other experiments. The Text Bag of Words + POS tags experiment produced close but less successful results to bag of words alone except for in pair 2 which was the Sandy Hurricane as source and the Boston Bombing as the target. We can also see that the error bars on most of the results are relatively small, thus the results were very close in value between each fold.

tags were used by themselves. There are too many on-topic and off-topic instances that share POS tags and thus do not provide enough difference to successfully classify new tweets.

How accurate is a classifier generated from the bi-grams of POS tags?

The classifications seemed to improve with bi-grams over the POS tags, I think this is due to the bi-grams being more unique between tweets than the singular POS tags and so it worked better. The number of instances that share the bi-grams between on-topic and off-topic is still too high and thus, the classification is not an improvement to the text alone.

2. Does knowing the part of speech of a tweet reveal new information previously unknown with just text?

   I would say no, it does not seem that knowing that a verb or a noun existing in a tweet informs us better than just text, when using POS tags as independent features. effectively label data.

3. Does the combination of text + POS tags result in improved classifications as compared to those labeled from either text or POS tags alone?

   Based on the results, it is clear that the resulting classifiers produced by text and POS tag features were less successful in comparison to the text features alone, but was more successful when compared to the the POS tag features by themselves. This is due to the fact that like previously stated both on-topic and off-topic tweets share too many POS tags and so when POS tags were added to the text only sets the new features acted more like noise than an aid and cause the generated classifier to be less successful.

   Is this combination better than the combination of text + POS tag bi-grams?

   I see an improvement across all the tests when the bi-grams were used instead of the

POS tags by themselves.

Chapter 6

Related Work

There has been research conducted in the field of text classification of tweets. One such experiment involving tweets and text classification was by Go et al. [2009] where they experimented with classifying tweets as positive or negative based on emojis within each tweet. The dataset they conducted their experiments on was scrapped from queries on the Twitter API over the period of April 6, 2009 to June 25, 2009. The scrapper looked for positive emoticons such as :) and :-) and negative emoticons such as :(and :-(to initially classify their source data. They then stripped out the emoticons so that the training algorithms they were using such as Naive Bayes, learned off of other features of a tweet. Their tweet preprocessing was near identical to that of our process. Once done preparing the data they had 800,000 negative tweets and 800,000 positive tweets. In their analysis, in one of their experiments they used POS tags as features. They came to the same conclusion as I did that POS tags did not produce a significantly positive result on their classifications.

Kouloumpis et al. [2011] also conducted a similar experiment involving POS tags of tweets. In their experiment, their test data consisted of tweets that contained hash tags and emoticons. The used a variety of different features in their classifications. They used a wide set of features such as features representing information about sentiment lexicon and POS

tags, with their base line being unigrams and bigrams. Their experiment was a two-fold cross validation setup. They wanted to see if derived labels from the two datasets improved the classifiers for tweets. They also reportedly did not receive successful results with POS tags and that they needed to conduct more research.

Studies done on twitter in regards to disaster management have increase in frequency in the past few years. Many of these studies attempt to mitigate the impact of a current disaster with the use of tweets. One study by Chatfield and Brajawidagda [2013], was on the impact of early twitter disaster warning systems. They conducted a content analysis of gathered tweets from all users who publicly tweeted as well as the twitter feed of the Meteorological,Climatological and Geophysical agency that was in charge of early warnings of disasters. They used specific keywords produced by this agency in the content analysis. They determined that although the system works at informing the public of pending disasters, its success is limited by the speed at which the system is able to inform the public by means of other agencies and the public retweeting the information.

There are some open source projects that attempt to collect and label disaster tweet data. One such program is the Artificial Intelligence for Disaster Response Platform (AIDR) published by Imran et al. [2014]. This platform was designed as a way for a user to filter a real time twitter stream for a given set of keywords and classify each tweet found. The labels for the classification are produced by crowd who label a sub-set of the filtered tweets. The classification of the tweets improves with time as more labeled data is received.

In the area of Domain Adaptation of tweets, Mejova and Srinivasan [2012] discussed the possibilities of generating classifiers that spanned across multiple data streams. In particular they experimented with generating classifiers of blogs, reviews, and tweets. Through their tests they found that the performance of some foreign domain classifiers were indistinguishable when compared to the performance of native domain classifiers. It was determined that based on the three different domains of information that reviews, which required little effort

in collecting and labeling, were the best at generating cross domain classifiers, then tweets, with blogs, which required the most effort, producing the least interesting classifiers.

It is important to note that the Domain Adaptation algorithm used in my experiments is not the first occurrence of its use. The algorithm was first developed by Tan et al. [2009]. They tested its effectiveness of classifying data across three domains of data, education reviews, stock reviews, and computer reviews which were all annotated by three linguists. In the end they were able to conclude that their approach provided better performance to some semi-supervised and transfer-learning techniques.

The algorithm was then take up by Herndon and Caragea [2013] for the purpose of classifying protein localization from the composition of the protein. Their findings were that the algorithm produced successful classifications in their experiment and that they found that the closer the target domain was to the source and the more of the target data that was labeled improved the performance even more.

The last group to use the algorithm before I used it was Li et al. [2015]. My experiments closely aligned with with theirs. They focused on using the algorithm to classify different disaster tweets. It was their starting data that I conducted my experiments with. For their data features, they used a bag of words of the tweet's text to construct their classifier. I analyzed some of the same source and target disaster pairs. In their experiments the results were successful for classifiers that used some data from the target than just from the source by itself when the disasters were in similar domains. If the disasters were in different domains then the classifier learned from the source was better due to the target data acting like noise and not being a helpful.

Based on my experiments and the result of prior research, the addition of POS tag unigrams and bigrams do not help as much in addition to text. Further refinements of the features, such as identifying dependencies, are needed to improve the performance obtained with text alone.

Chapter 7

Conclusion

7.1 Summary and Conclusion

There are many factors that can help or hinder a Naive Bayes classifier when it comes to classifying disaster tweets. In past experiments the features used were simply the bag of words representation of the tweet. Each word in the collection becomes its own feature to be used in the generation of a classifier. My experiments were an attempt to discover a new set of classifying features for tweets. Part of speech tags are features that can help enhance classifiers, when used with standard text documents. The idea of this work was that if a POS tag can add more information in standard text, then maybe it can do the same thing in tweets. To test this idea, I followed the work of Li et al. [2015], who had done similar work in regards to tweets and classification. They focused on the generation of a classifier from one source disaster, with the goal of successfully labeling tweets of a target disaster. I ran the same experiments with the same goal in mind and added POS tag unigrams and bigrams as new features. The POS tags were collected from the POS tagger tweboparser developed by Owoputi et al. [2013]. This tagger was specifically designed to analyze tweets. The five sets of experiments I ran resulted in the discovery that POS tags alone could not

produce accurate classifiers when compared to the classifiers produced from a bag of words representation. The bi-grams of the POS tags improved on the classification when compared to the POS tags alone, but were still not as successful as the text alone. When added to the bag of words features, the POS tag features seemed to overall decrease the performance of the classifier as well. Other prior research has reported results that confirm my findings that, at least with POS tags or bi-grams of POS tags, will not help much in addition to text. Further refinements are needed to improve the performance obtained with text alone.

7.2 Future Work

Based on the results of my experiments, I determined that the POS tag uni-gram and bi-gram features did not yield an improved classifier. Instead of using the those features to construct a classifier I would like to find and use a dependency parsing tool to analyze the structure of a tweet. To also round out my experiments I would also like to construct word bi-grams features of the text and generate a classifier based on those and combine them with the POS tag bigrams to see if I could produce better classifications compared to my other experiments.

Bibliography

Chatfield, A. T. and Brajawidagda, U. (2013). Twitter early tsunami warning system: A case study in indonesia's natural disaster management. In *System sciences (HICSS), 2013 46th Hawaii international conference on*, pages 2050–2060. IEEE.

Dale, R. (2000). *Handbook of natural language processing*. Marcel Dekker, New York.

Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., and Smith, N. A. (2011). Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.

Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1:12.

Herndon, N. and Caragea, D. (2013). Predicting protein localization using a domain adaptation approach. In *International Joint Conference on Biomedical Engineering Systems and Technologies*, pages 191–206. Springer.

Imran, M., Castillo, C., Lucas, J., Meier, P., and Vieweg, S. (2014). Aidr: Artificial intelligence for disaster response. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 159–162, New York, NY, USA. ACM.

Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145.

Kouloumpis, E., Wilson, T., and Moore, J. D. (2011). Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11:538–541.

Li, H., Guevara, N., Herndon, N., Caragea, D., Neppalli, K., Caragea, C., Squicciarini, A., and Tapia, A. H. (2015). Twitter mining for disaster response: A domain adaptation approach. In *the 12th International Conference on Information Systems for Crisis Response and Management, Kristiansand, Norway*.

McCallum, A., Nigam, K., et al. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer.

Mejova, Y. and Srinivasan, P. (2012). Crossing media streams with sentiment: Domain adaptation in blogs, reviews and twitter. In *ICWSM*.

Olteanu, A., Castillo, C., Diaz, F., and Vieweg, S. (2014). Crisislex: A lexicon for collecting and filtering microblogged communications in crises. In *ICWSM*.

Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., and Smith, N. A. (2013). Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47.

Tan, S., Cheng, X., Wang, Y., and Xu, H. (2009). Adapting naive bayes to domain adaptation for sentiment analysis. In *European Conference on Information Retrieval*, pages 337–349. Springer.

Zhu, X. (2005). Semi-supervised learning literature survey.

Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3:1130.