CORE
Metadata, citation and similar papers at core.ac.uk
Provided by University of Louisville

University of Louisville
ThinkIR: The University of Louisville's Institutional Repository

Electronic Theses and Dissertations

12-2018

A transfer learning approach for sentiment classification.

Omar Abdelwahab
University of Louisville

Follow this and additional works at: https://ir.library.louisville.edu/etd

 Part of the Other Computer Engineering Commons, and the Other Computer Sciences Commons

Recommended Citation

Abdelwahab, Omar, "A transfer learning approach for sentiment classification." (2018). *Electronic Theses and Dissertations.* Paper 3124.
https://doi.org/10.18297/etd/3124

This Doctoral Dissertation is brought to you for free and open access by ThinkIR: The University of Louisville's Institutional Repository. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of ThinkIR: The University of Louisville's Institutional Repository. This title appears here courtesy of the author, who has retained all other copyrights. For more information, please contact thinkir@louisville.edu.

A TRANSFER LEARNING APPROACH FOR SENTIMENT CLASSIFICATION

By

Omar Abdelwahab
MS, University of Louisville, 2013
BSc, Cairo University, 2010

A Dissertation Submitted to the Faculty of the J.B Speed School of Engineering
Of the University of Louisville in Partial Fulfillment of the Requirements for the Degree
of

Doctor of Philosophy in Computer Science and Engineering

Computer Engineering and Computer Science Department
University of Louisville
Louisville, Kentucky

December 2018

Copyright 2018 by Omar Abdelwahab

All rights reserved

A TRANSFER LEARNING APPROACH FOR SENTIMENT CLASSIFICATION

By

Omar Abdelwahab
MS University of Louisville, 2013
B.Sc. Cairo University, 2010
A Dissertation approved on

November 12th, 2018

By the following Dissertation committee

_____
Dissertation Director
Adel Elmaghraby

_____
Eric Rouchka

_____
Jeffrey Hieb

_____
Michael Losavio

_____
Ibrahim Imam

DEDICATION

To Mom, Karim, and Dad

ACKNOWLEDGMENTS

I would like to thank everyone who helped me through the tough times of the Ph.D. program. From Dr. Elmaghraby to my Ph.D. colleagues here at the Computer Engineering and Computer Science Department at the University of Louisville. I would like to thank my family for their continuous support throughout my life to this day. Special thanks to my friends TJ Singh Sethi, Harry Talamini, Olivia Trohler, Caleb Shehan, Yehya Senousey, Mohamed Abdelfadeel, Mehdi Sabraoui and Ahmed Magdy for their help and continuous support during my time in Louisville. I would also like to thank people who I got to know during a pivotal time in my PhD and who influenced me in many ways while I had the pleasure of working with them at REACH back in 2015 as a GSA and who will remain dear friends of mine Nathan Dalton, Caitlin Johnson, and Sara Heath. I would also like to thank Dr. Abdullkader of Voicera for helping me with his advice during his tenure with Facebook and Voicera. I cannot thank him enough for his time and helpfulness. I can certainly say that Dr. Abdullkader had a positive impact on me that will definitely help me as I go along with my career. I would like to thank Dr. Jeffrey Hieb for his helpful feedback and attention to detail. I would also like to thank Dr. Eric Rouchka for his feedback on my work during and after the proposal. Special thanks to Dr. Elmaghraby, my mentor for his support. Finally, I would like to thank Dr. Imam and Dr. Losavio for being an integral part of the committee.

ABSTRACT

TRANSFER LEARNING APPROACH FOR SENTIMENT CLASSIFICATION

Omar Abdelwahab

November 12th, 2018

The idea of developing machine learning systems or Artificial Intelligence agents that would learn from different tasks and be able to accumulate that knowledge with time so that it functions successfully on a new task that it has not seen before is an idea and a research area that is still being explored. In this work, we will lay out an algorithm that allows a machine learning system or an AI agent to learn from k different domains then uses some or no data from the new task for the system to perform strongly on that new task. In order to test our algorithm, we chose an AI task that falls under the Natural Language Processing domain and that is sentiment analysis. The idea was to combine sentiment classifiers trained on different source domains to test them on a new domain. The algorithm was tested on two benchmark datasets. The results recorded were compared against the results reported on these two datasets in 2017 and 2018. In order to combine these classifiers' predictions, we had to assign these classifiers weights. The algorithm made use of the similarity between domains when inferring the weights for the classifiers trained on the source domains by measuring the similarity between these source domains and the domain of the new task concluding, that domain similarity could be used in computing weights for classifiers trained on previous tasks/domains.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

CHAPTER I

INTRODUCTION

Transfer learning is an active machine learning research area that involves the use of labeled and unlabeled data samples from S source domains and few labeled or unlabeled samples in the target domain to help a machine learning system to adapt in order to perform well on the target domain. S could range from 1 to multiple source domains. Transfer learning is also referred to as cross-domain adaptation. There have been various transfer learning methods developed over the past couple of years on many applications from image processing and classification to numerous text classification tasks. One of the most popular text classification tasks that make use of transfer learning is sentiment analysis. There are other text classification tasks such as part of speech tagging (POS tagging) and spam opinion mining that make use of transfer learning. The purpose of this work is to show that classifiers trained on previous tasks or previous source domains can be combined together in an AdaBoost inspired way by using the domains similarities between the source domains and the target domain to derive weights for the source domain classifiers when aggregating their prediction outputs on the target domain. Secondly, it is to develop an unsupervised transfer learning system that uses labeled and unlabeled samples from the source domains and only unlabeled samples from the target domain without needing any labeled samples from the target domain as labeling training samples is an expensive task. Finally,

developing a lifelong learning system that uses no data from the target domain during training as we will describe later in the dissertation. The lifelong learning system was developed by adding a tweak to our transfer learning system. We have compared our results against the results reported by another lifelong learning sentiment classification system (Chen et al. (2018)) on the benchmark dataset provided by Chen et al. (2018). Some transfer learning techniques have resorted to domain similarity as a way to help the transfer learning process between different domains. However, to the best of our knowledge, none of the techniques published in the past decade have used domain similarities directly when adapting classifiers trained on different source domains. Domain similarities have been used in feature selection and in identifying certain features that might have different sentiment polarities in different domains as in Wu et al. (2016). Instead of combining training samples from different domains, we have developed a transfer learning-based algorithm that combines classifiers trained on S source domains for applying them to a target domain. The idea is to use the classifiers that have been learned from X previous tasks. Our algorithm computes the similarities between the source and target domains to infer weights to be associated with the classifiers trained on the source domains. Each classifier was given a weight that was derived from its domain's similarity to the target domain as we will show later in the methodologies chapter. The algorithm is inspired by the AdaBoost algorithm. Except that the weights for the source classifiers were computed differently from how the classifier weights were calculated in Adaboost. As in any unsupervised transfer learning task, we assume that we do not have labeled samples in the target domain. Therefore, it was not possible to use AdaBoost's method in computing the weights of the classifiers in the ensemble as they were computed using the classifiers'

training errors on the labeled training sets as we will show later in the methodologies chapter. In our case, we do not have a labeled training set in the target domain. So we have used the domain similarities to compute the weights of our source classifiers as we will show later on. We have called our method Adaboost Inspired Transfer Learning Approach for Sentiment Classification (ATLAS). In the next chapters, we will cover prior work in the literature review, the ATLAS method and other algorithms in the Methodology chapter, the results that we have recorded and how they compare against the results reported in Wu et al. (2017) and Chen et al. (2018) in the experimental results chapter and our conclusion and future work in our final chapter.

CHAPTER II

LITERATURE REVIEW

Several transfer learning/domain adaptation approaches have been proposed recently to reduce the accuracy loss in cross-domain sentiment classification. Approaches like spectral feature alignment (SFA), Structural correspondence learning (SCL) and graph-based algorithms such as RANKER and OPTIM. Ponomareva et al. (2012) compared between graph-based algorithms and the state of the art (SCL) and (SFA) algorithms where they concluded that the graph-based algorithms OPTIM and RANKER gave competitive accuracies when compared with SCL and SFA. In Aue and Gamon (2005), the authors experimented with four strategies to build sentiment classifiers to new target domains in the absence of large data in these domains. First, they trained a model on a mixture of labeled data from other domains, then tested their model on the target domain. Secondly, they trained the second model in the same way as the first but they limited the number of features used in training to those found in the target domain only. Thirdly, they trained ensembles of classifiers on different source domains with abundant labeled training data and tested it on the target domain. Finally, they developed a semi-supervised approach which uses small amounts of labeled data with large amounts of unlabeled data in the target domain. SVM was used for the first 3 strategies and Expectation Maximization for the 4th. The 4th approach gave the best performance as it was able to make use of both the

labeled and unlabeled data in the target domain. In Yang et al. (2006), a strategy based on selecting domain independent features from both domains was proposed. The method utilized fully labeled training sets from two domains to select highly ranked domain independent features from both domains and these features were later used in training the final classifier for the target domain. In Tan et al. (2009), a simple strategy was proposed where a base classifier is trained on the labeled data of the source domain and then the 10 classifiers are used to label some informative observations in the target domain. Using the selected informative observations in the target domain, a new classifier is learned which is then used to classify test cases from the target domain. Blitzer et al. (2007) proposed an approach called structural correspondence learning (SCL) developed by Blitzer et al. (2006) for transfer learning. Given labeled reviews from the source domain, unlabeled reviews from the source and target domains, SCL first selects J features that occur frequently in the source and target domains and that have large mutual information gain with the source labels in the source domain. These features are called pivot features. A correlation matrix W is then formed to measure the correlation between the pivot and non-pivot features in both domains where each row i represents the correlation between pivot feature i and all the other non-pivot features. Consequently, singular value decomposition (SVD) is applied to compute the left singular vectors transposed of W. The final features used for training and testing were a combination of the pivot features and the top k-x non-pivot features that have the highest correlation with the pivot features. So the final set will contain k features. Daume et al. (2007) proposed a frustratingly easy domain adaptation that some people refer to it as Easy Adapt. The approach is appropriate in the case of having labeled source and target domain data. The approach is so simple, it could be implemented

as a preprocessing step and it performs better than the PRIOR baseline f 11 Daume et al. (2010) that utilized labeled and unlabeled data in the target domain. Tan et al. (2009) added an improvement to the SCL algorithm by proposing a feature weighted and instance weighted SCL model, which weighs the features as well as the instances' polarity. The authors addressed the issue of having high-frequency domain-specific (HFDS) features that correspond with the pivot features when using SCL and how these features would decrease the influence of the original pivot features. Thus they have proposed a feature-weighted SCL to adjust the influence of HFDS features in building correspondence by assigning a larger weight for observations with the same sentiment polarity as the corresponding pivot features. Pan et al. (2010) proposed a method that works in the setting where there are only labeled examples in the source domain and unlabeled examples in the target domain. The algorithm uses a spectral feature alignment (SFA) algorithm to align domain-specific words from different domains into unified clusters. Domain-Independent words are like pivot words in SCL. SFA works by first constructing a bipartite graph with the domain-independent words as one set of nodes and the domain-specific words as the other set of nodes. A domain-specific word is connected to a domain-independent word if they both co-occur together in the same document or within the same window where the weights on these links are the frequency of co-occurrence of these words together. Then domain-specific words that have more connections with domain-independent words are clustered together to form a feature set, and the domain-independent words that have more connections with domain-specific words are clustered together in another feature set. Then the training and testing sets are represented in a combination of these feature set clusters created during the cross-domain adaptation process. He et al. (2011) extracted opinion

topics from both domains to link them. The resulting topics that cover both domains are then used as additional features to the original features created for classification. Glorot et al. (2011) proposed a deep 12 learning approach for domain adaptation which consisted of two steps. The first step involved using a system based on stacked De-Noising Auto-Encoders with sparse rectifier units for unsupervised feature extraction from 22 different domains which provided high-level features for the linear classifier trained in the second step for the target domain. Their approach outperformed two of the state of the art approaches SCL and SFA which will be discussed later in the methodologies chapter. Guerra et al. (2011) proposed a transfer learning approach for real-time Twitter sentiment analysis by predicting opinion holder bias towards a topic by analyzing users' retweets and endorsements and used this feature in combination with textual features to improve the overall accuracy of real-time Twitter sentiment analysis where a sufficiently labeled training data is not available. By integrating the bias learned from only 10% of users who commented about a specific topic, the authors were able to correctly classify the polarity of 80% to 90% of the tweets. Gong et al. (2013) proposed a new domain adaptation algorithm for sentiment classification and image classification. Their approach consists of three stages. The first stage is in extracting what they describe as "Landmark" features from the source domain that is somehow similar to the target domain. Afterward, these landmark features are added to the source and target domains to create new auxiliary domains from which the features for the original adaptation problem is extracted from. Afterward, the landmark feature labels (Landmark feature x: Sentiment Polarity) with the auxiliary domains' features are combined to extract discriminative domain invariant features that are later fed into the sentiment classifier. Andreevskaia and Bergler et al.

(2008) wanted to integrate the domain independent knowledge of a lexicon based classifier (LBC) and the domain dependent knowledge of a corpus-based classifier (CBC) to overcome domain independence by training two systems, a lexicon based classifier that uses lexicons such as WordNet (Fellbaum et al. (1998)) and fuzzy logic for sentiment classification and another corpus-based classifier that is 13 trained on a small sample of in-domain labeled data. The results show that combining a CBC and an LBC in an ensemble gives way better classification accuracies on various data sets from different domains than when only using an LBC or a CBC model. Zhou et al. (2010) tackled the problem of having a small amount of labeled training data while having abundant unlabeled data by developing a semi-supervised approach to sentiment classification called active deep nets (ADN). They first started with training the active deep net layers with greedy layer-wise restricted Boltzmann machines (RBM). They looped over the training set samples to calculate the weights of each layer in an ADN using two sigmoid functions. Then, the ADN was trained using the small labeled samples through gradient descent to minimize its loss function. Afterward, samples that had the smallest distances to the decision boundaries were selected for manual sentiment annotation and added to the labeled sample after which the whole process is repeated again for a specific number of iterations. The results show that their approach ADN outperformed or gave similar accuracies to other semi-supervised techniques such as TSVM, Active learning, DBN, MECH, and semi-supervised spectral learning. Kang Li and Zhao et al. (2009) proposed a domain adaptation method for sentiment analysis that consisted of two stages. The first stage is for feature translation by determining the common topics between the source and target domain as a bridge for the classifier to recognize the polarity distribution of the different domain-specific features that

describe these common domain independent topics. Then the second stage is for training a classifier on the source domain to classify unlabeled samples in the target domain to select some informative target samples to use for retraining the original classifier thus updating the original decision hyperplane till reaching a specific convergence threshold. Xia and Zong et al. (2011) proposed a POS-based ensemble approach for cross-domain sentiment classification. Since the authors tried to make use of their observation that some POS tags are domain dependent and some 14 are not. For instance, by efficiently utilizing the domain-independent POS-tagged features for words like 'love' and 'great' with ensemble classification, it would result in better performing cross-domain approaches when compared with using single classifiers that do not adapt on the target domain as shown in their study. Ponomareva and Thelwall et al. (2012) proposed an algorithm for automatic estimation of performance loss in the context of cross-domain sentiment classification. Factors like domain complexity are added for approximating performance loss when training a classifier on a source domain then testing it on data from a different target domain. Such algorithms help in deciding the size of the labeled target domain samples needed during the adaptation process as the amount of labeled target domain data is dependent on the similarity between the source and target domains as stated in Blitzer et al. (2006) and Blitzer et al. (2007). A comparative study between the graph based cross domain approaches and non-graph based approaches such as SCL and SFA were conducted by Ponomareva and Thelwall et al. (2012) and they have concluded that graph-based algorithms OPTIM and RANK consistently outperformed SCL and SFA for half of the cases. However, since the authors consider only the best accuracies obtained with RANK, such comparisons are not completely fair but it shows the potential of the RANK algorithm.

Wei and Pal et al. (2010) on the other hand use annotated English corpus for training a sentiment classifier to be tested on a corpus of another language which makes the adaptation problem in this case way harder. As the authors translate the target domain data then they resort to using reliable sections of the translations in addition to structural correspondence learning (SCL) for the adaptation problem. Bollegala et al. (2011) used multiple sources to construct a Sentiment Sensitive Thesaurus for cross-domain sentiment classification. The authors combined multiple source domains for training and they compared their results to other domain adaptation techniques such as SCL, SFA and LSA (which is based on latent semantic analysis). 15 The proposed solution beat all other previous approaches in three out of the four target domains tested. Scheible and Schutze et al. (2013) applied transfer learning (cross-domain adaptation) for the task of classifying sentences in a document as sentiment relevant text that affects the overall sentiment of a document or sentiment non-relevant text that has no impact on the overall document sentiment. The authors argued that transfer learning improves sentiment relevance classifications by 12 %. Li et al. (2013) proposed an active learning approach to cross-domain sentiment analysis where two classifiers were trained. One trained on labeled data from the source domain and the other on labeled data from the target domain. Then both classifiers were combined to select informative unlabeled samples from the target domain called uncertain samples for manual labeling by a strategy called Query by Committee (QBC). After updating the training data with the newly labeled samples a label propagation based graph algorithm was deployed to propagate the class labels from the labeled data to the unlabeled data. The results show that the proposed system outperformed SCL in seven out of the twelve adaptation tasks on the Blitzer et al. (2007) benchmark data set. In 2015,

Chen et al. (2015) have proposed a Lifelong learning algorithm for sentiment analysis. Where given k domains, you train a system using k-1 domains then applying the system on the kth domain without using any data from the kth domain during training as we will discuss later on. An updated version of Chen et al. (2015) paper published in ACL 2015 was published recently in January 2018 in Arxiv and we will be comparing our results against the updated version published in Chen et al. (2018). The authors have developed a system called LSC composed of four parts. The first part, was referred to as past information store (PIS) where they stored the results of previous tasks where for every word they would store the probabilities of each word given a positive or a negative label at task t Pt(w|+) and Pt(w|-) then the number of times each word w appeared in positive document and negative documents. The second part contained two types of information, document level knowledge (number of occurrences of word w in positive and negative documents) and number of past tasks where P(w|+) > P(w|-) and P(w|-) > P(w|+). The third component was a knowledge miner that performs counting and aggregation of the information collected in the PIS. Finally, the fourth component called knowledge-based learner that incorporates the knowledge collected using regularization techniques to optimize their algorithm's learning. The advantages of the LSC algorithm in Chen et al. (2018) is that it uses no labeled or unlabeled samples from the target domain. However, we were able to tweak our approach (ATLAS) to function as a lifelong learning system without the need of any samples from the target domain as we will show in the results chapter when comparing between our approach (ATLAS) and the algorithm proposed by Chen et al. (2018). Wu et al. (2016) proposed a Multiple domain sentiment classification system that was based on first measuring the similarities between different domains by extracting the

term distributions between domains then inferring the sentiment relationships between these words across different domains. As a positive word could have a negative leaning polarity when used in another domain which is referred to as the feature mismatch problem. After that step, their system trains their global model on the information extracted from the initial training phase. Wu et al. (2017) came up with another technique called ASDA which we will compare our model against. Where the authors have trained their model using multiple source domains and an additional global lexicon that contained general sentiment polarity data. We will be comparing our system ATLAS against ASDA as it is the latest multiple source domain adaptation for sentiment classification system published. We will also compare our results on a second benchmark dataset Chen et al. (2018) and compare our results to the results published in Chen et al. (2018).

CHAPTER III

METHODOLOGIES

In this chapter, we will go in-depth with explaining the methods that we have used in improving the cross-domain adaptation process for sentiment analysis when tested on four target domains of the Blitzer et al. (2007) benchmark dataset and 20 domains of the Chen et al. (2018) data set. Both data sets are publicly available.

We have developed the ATLAS algorithm to be an unsupervised transfer learning algorithm. Where if given K domains. The algorithm utilizes the labeled and unlabeled reviews of the K-1 source domains plus the unlabeled samples of the target domain. However, we have managed to add a tweak to our algorithm that enables it to be a lifelong learning algorithm. Where it gets to use the labeled and unlabeled reviews from k-1 source domains without using any data from the target domain as we will explain later in the chapter. We will cover the transfer learning ATLAS then, we will showcase the lifelong learning ATLAS, how the reviews were represented and the distance similarity techniques used.

TRANSFER LEARNING ATLAS OVERVIEW

The algorithm that we have developed is called the ATLAS algorithm which stands for Adaboost Inspired Transfer Learning Approach for Sentiment Classification. The idea is inspired by the Adaboost algorithm where you have classifiers that have better than random guessing performance combined together to boost the overall classification performance when tested on a test set. Each classifier in the ensemble is given a weight that is learned from its training error. Then a weighted sum of the classifiers' output is computed. If the weighted sum is above a certain threshold, a positive label is given to the test sample/input and a neg label is given otherwise. The weights are calculated based on the following equations:

$$Total\_weight = \sum_{i=0}^{len(dataset)} Dwi \quad (1)$$

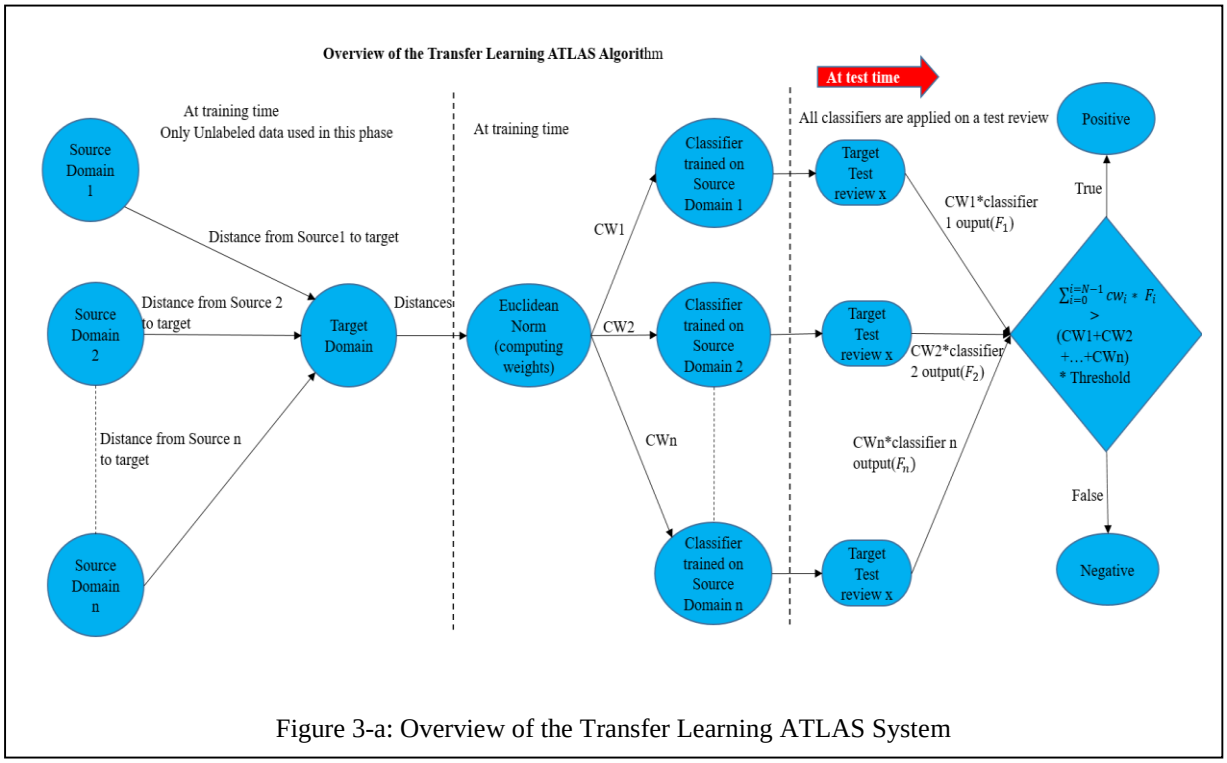

Figure 3-a: Overview of the Transfer Learning ATLAS System

Equation 1 represents how the total weights of all reviews in a single dataset is calculated. Each review/data point in a dataset has a weight that is equal to Dwi where i is the index of the review in the dataset. At the beginning of training, Dw is set randomly for all reviews and updated based on how many times a classifier classifies each review correctly. Therefore, reviews that get misclassified frequently get higher weight which will reflect on the weighted error of the classifier.

$$Total\_weight\_mistakes = \sum_{i=0}^{len(dataset)} Dwi * Correct\_prediction(\hat{P} == P) \quad (2)$$

Equation 2 represents how the total weight of mistakes is calculated. The correct_prediction function returns 1 if the predicted output is different from the gold standard while 0 otherwise. The return of this function is multiplied by the data point's weight and the computation is repeated across all points then all values are summed up to get the total weight of the mistakes which will be used in calculating the weighted error.

$$weighted\_error = \frac{Total\_weight\_mistakes}{Total\_weight} \quad (3)$$

Equation 3 shows how the weighted error for each classifier is calculated by dividing the total weight of the data points/reviews that were incorrectly classified by the total weight of all the data points/reviews in the data set.

$$Wc = 1/2 * ln(\frac{1-weighted_{error(c)}}{weighted_{error(c)}}) \quad (4)$$

Equation 4 shows how the final weights assigned to each classifier c in the ensemble is computed. The weighted error(c) represents the weighted error of classifier c.

Adaboost proved to be a powerful machine learning algorithm that has been widely applied to different problems. We have wanted to develop an Adaboost inspired transfer learning

algorithm for sentiment classification. As a result, we have chosen the 4 main domains that were used in Bollegala et al. (2015) and various other ACL papers of the Blitzer et al. (2007) dataset. Then we randomly picked 10 additional domains included in the Blitzer et al. (2007) dataset to implement and test our ATLAS algorithm on. The total number of domains included were 14. We have tested our algorithm on the four main domains mentioned in Bollegala et al. (2015) paper and in the Wu et al. (2017) paper. The ATLAS idea is based on given labeled and unlabeled samples of k-1 domains and unlabeled samples of the kth target domain, k-1 classifiers are trained on their respective k-1 domains. Then the domain similarity of each of the k-1 source domains to the kth target domain is computed. The domain similarity distances recorded for the k-1 source domains are then normalized and used as weights to the k-1 classifiers that were trained on the k-1 domains. For example, if domain 1 has a distance x to the target domain, then the classifier trained on domain 1 will have a weight that is equal to the normalized value of x. All k-1 weights are calculated by computing the Euclidean norm of all the k-1 distances. Since Adaboost calculates the weights associated with each classifier in the ensemble by computing its training error, The ATLAS algorithm calculates the weights associated with the k-1 classifiers by computing the similarities between the k-1 domains to their target domain as there are no labeled training samples in the target domain. Assuming that classifiers trained on similar domains to the target domain will perform better on the target domain as opposed to classifiers trained on domains that are far away from the target domain. We will start with covering our data representation methods, then the distance similarity metrics and the in-depth details of the ATLAS algorithm where we will show how we aggregate the outputs from the k-1 source classifiers when applying them on the target domain.

DOCUMENT REPRESENTATION METHODS

We have represented each domain with three different representation methods. Term distribution, Average doc2vec vectors, the mean of the average word2vec vectors in all reviews in a specific domain. We will first cover the term distribution representation first then will cover the word2vec and doc2vec representations. The term distribution representation was straightforward and simple. The terms that had the Verb, Adverb, Adjective and Noun part of speech tags and a WordNet (Fellbaum et al. (1998)) positive or negative sentiment score of greater than 0.8 were counted. A dictionary of term counts was recorded for each domain. We have tweaked this representation by counting the frequencies of all Verbs, Adverbs, Adjectives, and Nouns in each domain without using a prior sentiment score knowledge as we will show in our second set of experiments on the Chen et al. (2018) benchmark dataset. The Word2Vec representation was introduced by Mikolov et al. (2013) and the doc2vec representation was introduced by Mikolov et al. (2014). The idea behind word2vec centers around learning numerical vectors of all words in a vocabulary that occur in a specific text corpus where words that co-occur together often are represented by vectors that are closer to each other in a vector space. The ideal word2vec model would assign word vectors to words such that when subtracting the word vector of the word “man” from the word vector of the word “King” then add the word vector of the word “Woman” get the word “queen”. We have trained our word2vec and doc2vec models on the reviews from all 14 domains mentioned in Table 3-a. However, since the book reviews domain has a large dataset size compared to other domains. We have decided to cap the maximum number of samples for us to use from each domain when training our word2vec and doc2vec models to 25000 reviews. Therefore, limiting the

Table 3-a: The number of positive, negative and unlabeled samples of the Blitzer et al. (2007) dataset

Domain	Positive Samples	Negative Samples	Unlabeled Samples	Total Samples
DVD	1000	1000	122438	124438
Books	1000	1000	973194	975194
Kitchen	1000	1000	17856	19856
Electronics	1000	1000	21009	23009
Apparel	1000	1000	7252	9252
Automotive	584	152	0	736
Baby	1000	900	2356	4256
Beauty	1000	493	1391	2884
CameraPhoto	1000	999	5409	7408
ComputerVideo	1000	458	1313	2771
Gourmet	1000	208	367	1575
Grocery	1000	352	1280	2632
Healthpersonal	1000	1000	5225	7225
JewelryWatches	1000	292	689	1981

influence of one domain on the doc2vec model. Table 3-a shows the number of positive, negative and unlabeled samples in each of our 14 domains. Word2Vec models are trained such that given the word vectors (which were randomly initialized at the start of training the word2vec model) of the 2k word window (t-k and t+k) around a target word t in a corpus that the word vectors of these words gets updated so that they get to predict the word vector of the target word by maximizing the average log probability in equation 5. The following equations are from Mikolov et al. (2014).

$$\frac{1}{T}\sum_{t=k}^{T-k} \log P(w_t | w_{t-k}, \dots\dots, w_{t+k}) \quad (5)$$

$$P(w_t | w_{t-k}, \dots\dots, w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \quad (6)$$

The denominator of equation 6 represents the summation of the exponential of the un-normalized log probabilities of all output words. While the numerator represents the exponential of the un-normalized log probability of word t. The value of y is calculated according to the following multi-class classifier like softmax which is represented by the following equation from Mikolov et al. (2014):

$$y = b + Uh(w_{t-k}, \ldots\ldots, w_{t+k}; W) \quad (7)$$

Where the U and b are softmax parameters and h is the average of the word vectors of the words surrounding the target word w_t from a set of all the weight vectors of all the words in the vocabulary W. The Doc2Vec vectors which are referred to as paragraph vectors were inspired from the word vectors mentioned earlier according to Mikolov et al. (2014). The idea behind word vectors is given k number of words in a specific context to predict the next word given the word vectors of the previous k words in the context. Paragraph/doc2vec vectors are tasked with predicting the next word given many sequences sampled from the paragraph. A paragraph could be a product review or an article or a tweet. Similar to word vectors, Doc2vec Models are trained such that each paragraph gets a unique vector, then the word vectors of all the words in a paragraph/review are averaged with the paragraph vector to predict the next word in a context. The resulting vector is then used as a unique vector for representing the paragraph/review. Training the paragraph vectors is similar to word vectors except that in equation 7, instead of averaging the word vectors only, the paragraph vector is also averaged with the word vectors. This model is called the distributed memory model of paragraph vectors. As the paragraph vector that is averaged with the word vectors acts as memory in capturing the context of the review/paragraph. The paragraph vector is shared across various contexts/sequences sampled from the same paragraph but not across paragraphs. Therefore, the paragraph vector gets updated as more contexts get samples from the paragraph and not just for one sequence in the paragraph thus capturing the context of the paragraph/review. Equation 5 represents the multiclass classifier/probability function that needs to be maximized for the word embedding to be learned. We have trained a 200 dimensional and 400 dimensional word2vec models in addition to a 200 dimensional and a 400 dimensional doc2vec models.

These 4 models were used in developing four ways for representing the 14 different domains. The result of these representations will be discussed in more detail in the results chapter. Here, we will show how these models were used in representing our domains. We have experimented with representing our domains using 200 dimensional word2vec vectors, 400 dimensional word2vec vectors, 200 dimensional doc2vec vectors, and a 400 dimensional doc2vec vectors. When representing a specific domain using the 200 dimensional word2vec model, we had to iterate over each review in the domain then we have calculated the 200 dimensional word vector of each word in the review then we have averaged all the word vectors in the review to get an average word2vec vector for each review. After calculating the average word2vec vector of each review in a domain. The average of all the average word2vec vectors in the domain is computed to get the mean average word2vec 200 dimensional vector that represents the domain. The same process is repeated when using the 400 dimensional word2vec model which lead to a 400 dimensional mean average word2vec vector. On the other hand, when using the doc2vec model. The doc2vec vector of each review in a domain is calculated using a doc2vec model. The trained doc2vec model takes a complete review as input and outputs one 200 dimensional vector in case of using the 200 dimensional doc2vec model and 400 dimensional vector in case of using the 400 dimensional doc2vec model. Then, all the doc2vec vectors of all reviews in a domain are averaged to get the average doc2vec vector that represents that domain.

DISTANCE METRICS

After representing our domains using the term distribution, Word2vec and Doc2vec representation methods. The second task was to measure how far these domains were from each other. Before aggregating or applying the ATLAS algorithm on the target test set, we measured the distances from the source domains to the target domain. These distances were then normalized using the Euclidean norm function to get the weights associated with the source classifiers. A classifier trained on a domain with a distance d and a normalized distance w to the target domain will have a weight equal to w associated with it when performing the ATLAS algorithm. We have used two distance measures in our ATLAS implementation, the cosine distance and the Euclidean Distance. We have used Low et al. (2012) for computing the cosine and Euclidean distances in addition to using their graphlab library in training our models. It is important to note that when measuring the distance between any source and target domains that were represented by the term frequency dictionary representation, the keys that were not found in one of the two dictionaries were considered to have a value of zero. For example, if a word such as “great” is in the source dictionary and not in the target dictionary then it is considered to be present in the target domain’s dictionary with a value of zero. Which implies that there are zero occurrences of that word in the target domain. In case of using the average doc2vec or mean average word2vec representations, the vectors in the source and target domains were treated as numerical lists of equal size so we did not have the same missing key issue as in the term frequency representation. Given two dictionaries or lists of equal lengths, the following equation shows how the cosine distance was calculated not the cosine similarity. Assuming

that the inputs x and y have d distinct variables. The output is a float number that represents the distance between the two input vectors.

$$Cosine\_D(x,y) = 1 - \frac{\sum_i^d x_i y_i}{\sqrt{\sum_i^d x_i^2}+\sqrt{\sum_i^d y_i^2}} \quad (8)$$

The following equation shows the Euclidean distance equation for calculating distances between dictionaries and lists of equal lengths.

$$Euclidean\_D(x,y) = \sqrt{\sum_i^d (x_i - y_i)^2} \quad (9)$$

d is the number of variables in each of the input vectors x and y.

After calculating the Euclidean and Cosine distances from each source domain to the target, we normalized them using the Euclidean Norm function below then used the normalized distances from this function as weights for our source domains' classifiers. The Euclidean norm is also referred to as the Frobenius norm. It is equal to the square root of the summation of the squares of its elements. The distances were normalized by dividing them by the Euclidean Norm which is calculated according to equation 11.

$$Euclidean\_Norm(x) = \sqrt{\sum_i^d (x_i)^2} \quad (10)$$

$$x_{normalized} = \frac{x}{Euclidean\_Norm(x)} \quad (11)$$

$$CW = 1 - x_{normalized} \quad (12)$$

After normalizing the Euclidean and Cosine distances using equations 10 and 11, they were then substituted within Equation 12 to compute the weight vector. Each normalized distance in the $x_{normalized}$ vector was subtracted from 1 such that classifiers that were trained on domains that were closer to the target domain (smaller normalized distance to target) will have a larger weight when compared to other classifiers that were trained on domains

that were further away from target. The next step was to aggregate the outputs of the source classifiers using the weights calculated in equation 12. The following equations show how the output of these classifiers were aggregated. Given n source classifiers numbered from 1 to n. The following equation shows how the weights calculated in equations 8 through 12 were utilized.

When training each logistic regression classifier on its corresponding source domain, the feature columns that are used in training the classifier are the unigrams, bigrams, trigrams, and tfidf. The unigrams are the counts of the single words in a sentence. While the bigrams are the counts of all the 2 word phrased that occur in the sentence and trigrams are the counts of all the 3-word phrases that occur in a sentence. The tfidf which stands for term Frequency-Inverse document frequency is a measure of a word w's local frequency vs w's global rarity in a sentence. Which is equal to the term frequency in a document/review D multiplied by the log of the total number of reviews/documents in a domain divided by the number of documents that word w appeared in. Equation 13 shows the tfidf calculation given word w and review d.

$$tfidf(word, d) = tf(word, d) * \log(\frac{N}{f(word)}) \quad (13)$$

The tf(word,d) term represents the frequency of word w in document/review d. While N represents the total number of documents/reviews in a domain. The f(word) term in equation 13, represents the number of documents in the domain that contain the word w while N is the total number of documents/reviews in the domain. After the tfidf is computed for each labeled training set in each domain along with the other features mentioned earlier (1-grams, 2-grams, 3-grams), we have trained a binary logistic regression classifier for each domain using the following equations. These four feature columns were translated to

thousands of unpacked features. These features were used in calculating the score in equation 14 that was substituted within equation 15 for calculating the probability P(Y=1|F1, F2,…Fn) of a positive label given all the unpacked features taken from the four feature columns (1-grams, 2-grams, 3-grams, tfidf).

$$Score = (weight_1 * f_1) + (weight_2 * f_2) + \cdots + (weight_n * f_n) \quad (14)$$

$$f_i(Weights, Features) = P(Y = 1|f_1, f_2, \dots, f_n, weight_1, weight_2, \dots . weight_{2n}) = \frac{1}{1+e^{-Score}} \quad (15)$$

Where W stands for all the weights associated with all the features used in training the logistic regression classifiers. The training process is performed using the following loss function where n is the number of samples iterated on per single iteration over the training set. The algorithm used for optimization was stochastic gradient descent. The first term in equation 16 $(y_i - f_i(Weights))^2$ represents the difference between the true label and the classification probability of the logistic regression model which stands for the residual sum of squares error (RSS). While λ_1 and λ_2 are regularization parameters that represent the L1 and L2 penalties that are used in decreasing/penalizing the values of our feature weights to avoid overfitting.

$$\min \sum_{i=1}^{n} (y_i - f_i(Weights))^2 + \lambda_1 \left|\left|Weights\right|\right|_1 + \lambda_2 \left|\left|Weights\right|\right|_2^2 \quad (16)$$

After training the logistic regression classifiers using equations 14 through 16, we have used the class weights (CW) calculated earlier in equation 12 in aggregating the outputs of the 13 source domain classifiers on the target domain's test set. Equation 17 shows the feature columns of the test set that were used at test time.

$$input\_review_i = test\_review_i[tfidf, 1-gram, 2-gram, 3-gram] \quad (17)$$

For each review in the test set, the final weights of each classifier in the ensemble were used in predicting the label of the review given the features of the input review. Afterward,

the output of the classifier was multiplied by its classifier weight calculated earlier in equation 12. The classifier only outputs 1 for a positive label and 0 for a negative label.

$$ATLAS_{Output}(F, CW, input_{review_i}) = \sum_{i=0}^{i=N-1} cw_i * F_i(Weights, Features\_input\_review) \quad (18)$$

If $ATLAS_{Output}(F, CW, input_{review_i}) > threshold * \sum_{j=0}^{j=n-1} cw_j$ then the review will be classified as positive otherwise the review will be classified as negative. The threshold value could be tuned during training by varying its value between 0.5 and 0.9 and recording the threshold value that resulted in the best performance on the validation set (not the test set). We will show in the results section that a threshold value between 0.6 and 0.7 resulted in the best accuracies and f1-scores recorded. After applying equation 18 on all test reviews, the ATLAS output was recorded and the accuracies and F1_Scores are shown in the results section.

LIFELONG LEARNING ATLAS

We have added a slight tweak to our ATLAS algorithm to function as a lifelong learning machine learning system instead of a transfer learning machine learning system. A transfer learning system learns from k-1 source domains and makes use of some unlabeled samples from the target domain. While a lifelong learning system learns from k-1 domains without using any labeled or unlabeled samples from the target domain during training. So instead of using the target domain's test set in addition to some unlabeled samples from the target domain when measuring the distances from all source domains to the target domain in equations 8 and 9, we have only measured the distance at test time between each test review to all labeled and unlabeled samples of the 13 source domains. The downside of this approach was the considerable increase in the time taken by the ATLAS system in classifying all test reviews as the distances (and therefore the classifier weights CWs) were re-computed with every test review. We have tested this tweak on the Chen et al. (2018 benchmark dataset and the accuracies in addition to the F1scores are shown in the experimental results chapter. The equation we used in calculating the F1Score is shown below:

$$Precision = \frac{T_P}{T_P + F_P} \quad (19)$$

$$Recall = \frac{T_P}{T_P + F_N} \quad (20)$$

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (21)$$

The precision is equal to the sum of the true positives divided by the sum of the true positives and the false positives. While the recall is computed by dividing the sum of the true positives by the sum of the true positives and the false negatives. The F1Score is a function of the precision and recall as shown in equation 21.

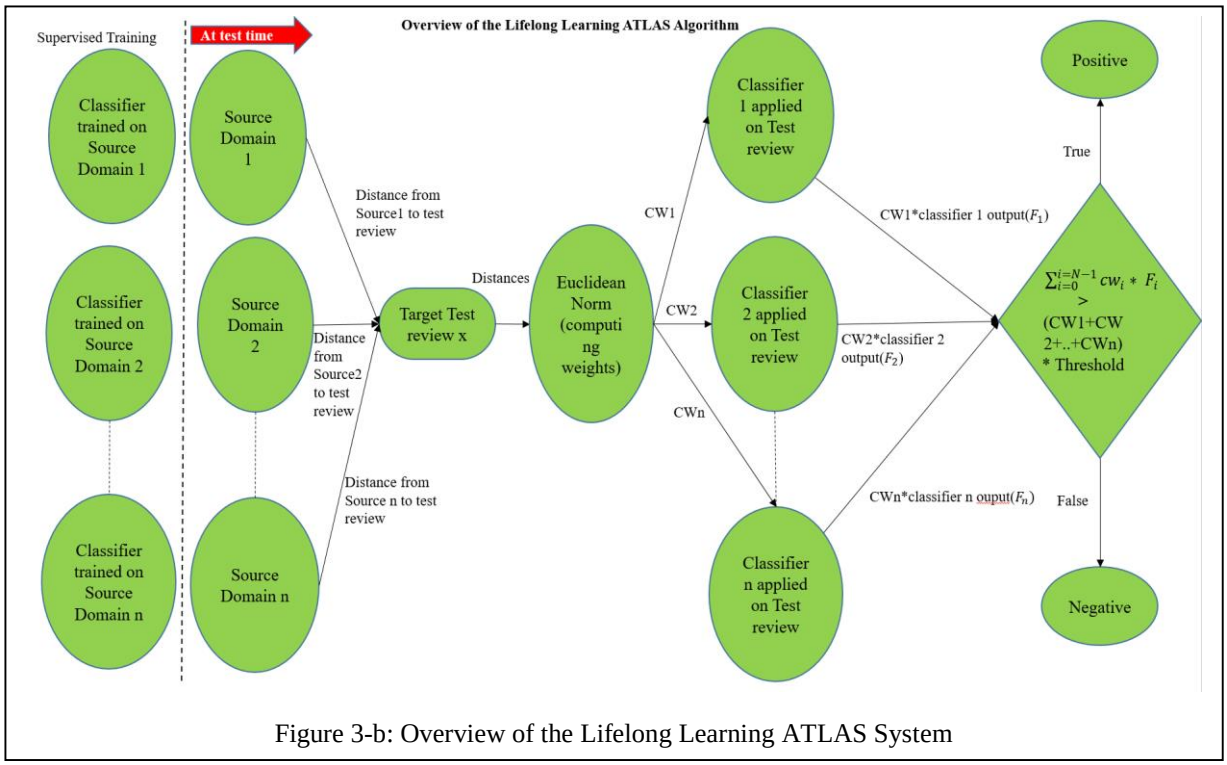

Figure 3-b: Overview of the Lifelong Learning ATLAS System

DEEP TEXT GENERATION

In this section, we will discuss a small experiment that is not related to our proposed ATLAS approach that we have come up with to test the effectiveness of deep learning based text generation for our transfer learning problem. The results we recorded did not encourage us to move along that route which led us to develop the ATLAS method. Here we wanted to show some of the text generation methods we have experimented with to generate labeled samples in the target domain for learning a target domain classifier. We have tested the text generation approaches on the kitchen domain of the Blitzer et al. (2007) and we have included the results in the experimental results section. The system consisted of three parts as shown in Figure 1. The first part focused on the rule-based sentiment labeling of the unlabeled kitchen product reviews of the Blitzer et al. (2007) data set. The

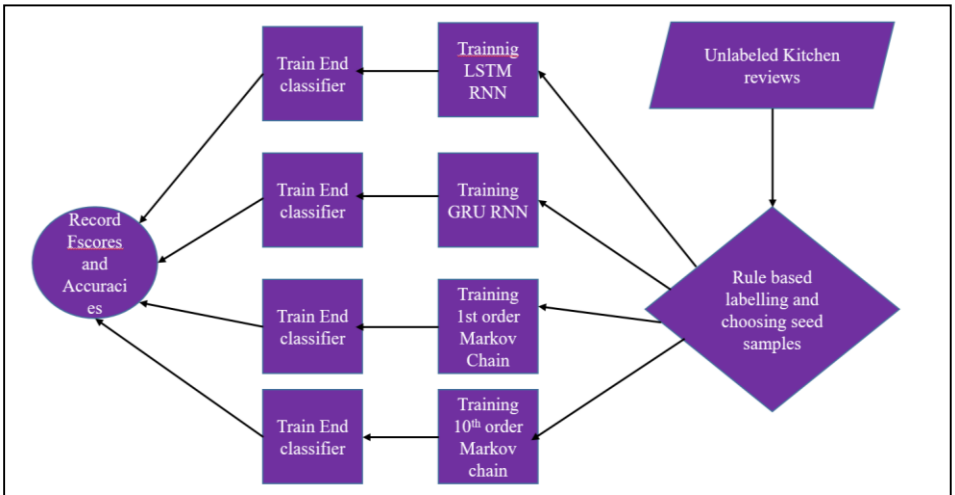

Figure 3-c: Deep text generation - System Overview

high confidence positive and high confidence negative labeled samples were chosen and named seed samples. The second part involved training deep learning based recursive

neural networks (LSTM and GRU), and Markov chain based text generators on the seed samples for generating positive and negative kitchen reviews. Finally, the third component focused on training a logistic regression model on the generated positive samples, negative samples and seed samples combined. The unlabeled kitchen product reviews were preprocessed according to Abdelwahab et al. (2015). In addition to removing stop words and replacing positive bearing ngrams with a "positive" symbol and similarly replacing negative bearing ngrams with a "negative" symbol. We have then used a simple rule-based technique in labeling the target domain samples then selected the high confidence samples from these labeled samples to be used for training the language models/text generators. The purpose of using a simple rule-based labeling technique for labeling a small sample of the target domain reviews was to compare the tolerance of the deep learning based text generation techniques against that of the Markov chain based techniques when being trained on a data set that is not 100% accurate. We will evaluate the performance of each text generation technique based on the F1Scores and accuracies achieved by the end classifier when tested on the benchmark kitchen test set (Blitzer et al. 2007) after being trained on the data generated from each text generation technique separately. The rule-based labeling algorithm is formed of the following steps. For each word in a review, a positive polarity score was calculated using the WordNet electronic Library's (Fellbaum et al. (1998)) pos_score function. Unigrams that had no pos_score or neg_score were assigned 0. The positive polarity scores were summed up then divided by the number of unigrams that had pos_score to get the average positive polarity score for the whole review, which was then stored. For each word in a review, a negative polarity score was calculated using the WordNet (Fellbaum et al. (1998)) Library's neg_score function. Unigrams that

had no neg_score were assigned 0. The negative polarity scores were summed up then divided by the number of unigrams that had neg_score to get the average negative polarity score for the whole review which was then stored. If there were no negative polarity ngrams in the review, the average negative polarity score was set to zero. Likewise, if there was no positive polarity bearing ngrams in the review, the average positive polarity score was set to zero. The average negative polarity score was subtracted from the average positive polarity score to get the polarity score difference between the average positive and average negative scores. If the difference was greater than +0.1, the review was labeled positive and if the difference was less than -0.1, the review was labeled negative. Reviews that had a polarity score difference in between -0.1 and 0.1 were labeled as unknown and were not used in training the language models. The reviews that were given a positive or a negative label after the rule-based labeling will be referred to as the "seed reviews" throughout the paper. The seed reviews were used in training the LSTM and GRU RNN models for text generation as will be shown in the next section. The following equations illustrate how the polarity score was calculated for each review. The term rev[i] in equations 1 and 2 stands for ith review in the data set.

$$AvgP = \frac{1}{p}\sum_{i=0}^{len(review)} pos\_score(rev[i]) \quad (22)$$

$$AvgN = \frac{1}{n}\sum_{i=0}^{len(review)} neg\_score(rev[i]) \quad (23)$$

$$Polarity_{Score} = AvgP - AvgN \quad (24)$$

TRAINING GRU, LSTM AND MARKOV CHAIN MODELS

We have experimented with varying the hidden state vector size and the number of layers of a GRU RNN and LSTM RNN networks. We have experimented with hidden state vector sizes of 64, 128, 256 and 512. Afterward, we kept the hidden state vector size at 50 then varied the number of layers from 2 to 5 then 10. Which lead us to train one GRU RNN model and one LSTM RNN model per hidden state vector size per polarity (positive text or negative text). Then one GRU RNN model and one LSTM RNN model per number of layers per polarity. While we varied the number of layers and the hidden size, we have set the number of epochs to 2000, learning rate to 0.01, chunk length to 200 and batch size to 100. The GRU and LSTM implementation were based on Robertson et al. (2017) implementation of the Character level text generation using GRU and LSTM. We have trained four Markov chain text generators. One Markov chain positive text generator of order 1, Markov chain negative text generator of order 1, Markov positive text generator of order 10 and Markov chain negative text generator of order 10. Each model was used in generating a balanced dataset of 100,000 positive and negative reviews for training the end classifier. The purpose is to compare using text generated by Markov chain generators that were trained on poorly unsupervised labeled seed samples against deep learning based text generators by supplying the data generated by each technique to the same end classifier and comparing the accuracies and F1Scores achieved when using the Markov chain based generated text vs the deep learning based generated text. The following equation represents the Markov chain model with order m. Where n>m and m was set to 1 then to 10.

$$Pr(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, .., X_1 = x_1) = Pr(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, .., X_{n-m} = x_{n-m}) \quad (25)$$

GENERATING TEXT AND TRAINING THE END CLASSIFIER

After varying the number of RNN layers and feature vector size we have ended up with 14 GRU models and 14 LSTM models. Each model generated 25,000 product reviews which made the total number of GRU generated reviews and LSTM generated reviews to 350,000 each. The 350,000 reviews consisted of 175,000 positive reviews and 175,000 negative reviews. After generating the positive and negative reviews, a logistic regression classifier was trained on the generated text combined with the seed samples and tested. Each generated training set used in training the model was parsed into a graphlab SFrame. Afterward, each review was pre-processed as in Abdelwahab et al. (2016) then a TFIDF calculation is made for each review in the SFrame. The logistic regression model was trained on the TFIDF column which resulted in hundreds of thousands of unpacked features where we had to use L1 and L2 regularization to do dimensionality reduction and to avoid overfitting. We have performed a grid search to find the near optimal L1 and L2 penalty values for the logistic regression model. The values taken by the L1 or L2 penalty variables during grid search were exponentially distributed. The graphlab library was used for training the logistic regression model. The following equations represent how the model was learned. Given a set of features xi, and a label yi∈{0,1}, logistic regression interprets the probability that the label is in one class as a logistic function of a linear combination of the features.

$$f_i(\theta) = P(y_i = 1|x) = \frac{1}{1+ e^{-\theta^T x}} \qquad (26)$$

The objective function tries to minimizes the output of the sigmoid function in equation 4 while adding the two regularization terms l1 and l2 adding the two regularization terms l1 and l2 penalties using mini batch gradient descent. Where θ is the weight matrix.

$$min_{\theta} \sum_{i=1}^{n} f_i(\theta) + \lambda_1 ||\theta||_1 + \lambda_2 ||\theta||_2 \qquad (27)$$

MARKOV CHAIN TRAINING ON SMALL SAMPLE OF LABELED TARGET DOMAIN DATA

We have trained additional Markov Chain text generators on 10% (160 positive reviews and 160 negative reviews) of the target domain labeled kitchen dataset in Blitzer et al. (2007) for the purpose of highlighting the major improvement in the performance of the Markov Chain text generators when training it on a small sample of 100% correctly labeled data samples. We have trained two Markov chain (one model with order 1 and the other with order 10) positive text generators and similarly two Markov chain based negative text generators. Each generator produced 50000 reviews. We have combined the positive and negative reviews generated by the order 1 generators into one data set and named it Markov_Labeled_1 and likewise, we have combined the positive and negative reviews generated by the order 10 models into one dataset called Markov_Labeled_10. We have added the 10% labeled reviews to each of these two data sets and supplied the end data sets to the end classifier. The results that we got show clearly that there was a double-digit improvement in accuracy and F1Score when using text generated from Markov chain text generators trained on 100% correctly labeled samples than using text generated from Markov chain text generators that were trained on a larger data set that was labeled by a simple unsupervised rule-based labelling technique that has an accuracy of 71% on the Blitzer Kitchen test set. The size of the weakly labeled (rule-based labeled) seed samples that were used in training the Markov Chain text generators was 33567, which consisted of 8093 negative labeled reviews and 25474 positive labeled reviews.

CHAPTER IV

EXPERIMENTAL RESULTS

In this chapter, we will showcase our results for our Adaboost Inspired transfer learning approach for sentiment analysis ATLAS. We have used two benchmark datasets. The Blitzer et al. (2007) and Chen et al. (2018). We will start first with explaining our results on the Blitzer et al. (2007) dataset then will discuss our results on the Chen et al. (2018) dataset. There are 24 different domains in the Blitzer et al. (2007) dataset. Most papers use only four domains to test their algorithm on. These domains are the DVD, Electronics, Kitchen, and Book product reviews. We have used these domains and we have randomly selected 10 additional domains from the Blitzer et al. (2007) to be included in our experiments. Table 3-a shown earlier outlines the domains included in our experiments with the number of their positive, negative and neutral reviews. Our algorithm as explained in the Methodologies section uses labeled and unlabeled data from the k-1 source domains and only unlabeled data from the kth target domain. The unlabeled reviews were mainly used in measuring domain similarities between different domains.

RESULTS OVERVIEW

We have started with the term frequency representation to represent each domain with its most occurring unigrams then we have alternated between the following domains Kitchen, DVD, Books, and Electronics by setting each one of them as the target domain and the rest of the remaining 13 domains were used as the source domains for training. We have started with training each classifier on its own domain first then recorded its test accuracy on the target domain's test set to show how each of these classifiers would perform if applied on the target test set directly. Each classifier's test accuracy on the test set was never used in training our system or in even ranking our classifiers in any way. For each domain, we have tabulated all of our results on the test set provided by Bollegala (2015) that was sampled from the Blitzer et al. (2007). Moreover, we have generated five random balanced test sets generated in the same manner as in Wu et al. (2017) from the Blitzer et al. (2007) then applied each of the source domain classifiers on the five randomly generated test sets in the target domain. Afterward, we have averaged the accuracies on the five test sets for each of the four target domains and we have shown the results in the in-depth analysis section. We will start with our algorithm's performance on the Bollegala et al. (2015) test sets that were sampled from the Blitzer et al. (2007). The following tables show the best results achieved on the Bollegala et al. (2015) exact test sets that consisted of 200 positive samples and 200 negative samples for each target domain. Tables 4-a and 4-b show the best accuracies achieved by the ATLAS algorithm when varying the classification threshold from 0.5 to 0.9. Table 4-c shows the best accuracies when using the Euclidean distance while Table 4-d shows the best accuracies recorded when using the Cosine distance. The Threshold row shows the classification threshold that contributed to

achieving the best accuracy. As for the classifiers row, we will show in the in-depth analysis that the source classifiers trained on the top n/2 closest domains to the target domain are called top performing classifiers while the classifiers trained on the top n/2 furthest domains to the target domain are called least performing classifiers. These titles are not based on the performance of the individual source classifiers on the target domain's test set in anyway. After identifying the top performing and least performing classifiers. We have formed three groups of ensembles to experiment with. The first ensemble contains the top performing classifiers only and called top performing. The second is called least performing group which contains the least performing classifiers only and the third is called the top_least performing group that contains both, the top performing and least performing classifiers. Therefore, the classifiers row shows which group of classifiers resulted in the best accuracy achieved on the target test set. The best accuracy results in tables 4-a and 4-b are better than the results presented in Bollegala et al. (2015) which is understandable. As Bollegala et al. (2015) performed transfer learning from one source domain to the target. However, our approach is a multi-source domain to target. As a result, we wanted to compare our results with another multi-source domain transfer learning system like the one presented in Wu et al. (2017). Tables 4-c and 4-d show the best average accuracies recorded when testing the ATLAS system on the five randomly generated test sets and comparing our results to a similar multi-domain transfer learning technique, the ASDA outlined in Wu et al. (2017).

Table 4-a: The best accuracies recorded when using the ATLAS algorithm on the exact Bollegala test sets that were sampled from the Blitzer et al. (2007) dataset.

Highest Cosine Accuarcy on Bollegala's Blitzer test set				
Blitzer et al. (2007) Dataset				
Domain	Kitchen	Books	Electronics	DVD
Best Accuracy	0.874	0.73	0.8275	0.7775
Distance Metric	Cosine	Cosine	Cosine	Cosine
Threshold	0.6	0.6	0.6	0.6
Classifiers	Top	Top	Top_Least	Top_Least

Table 4-b Best Accuracies when using the Euclidean distance on the Bollegala test sets (sampled from Blitzer et al. (2007))

Highest Accuracies on the Bollegala's Blizter test set in general				
Blitzer et al. (2007) Dataset				
Domain	Kitchen	Books	Electronics	DVD
Best Accuracy	0.875	0.745	0.83	0.7825
Distance Metric	Euc	Euc	Euc	Euc
Threshold	0.6	0.6	0.6	0.6

The results achieved beat the results reported by Bollegala et al. (2015). However, we are not comparing our algorithm against Bollegala et al. (2015). As their algorithm was not a multi-source cross domain adaptation algorithm. It was adapting a sentiment analyzer from one source domain to one target domain. We will compare our algorithm (ATLAS) against Wu et al. (2017) ASDA algorithm when using the same benchmark dataset, they have experimented with (Blitzer et al. (2007)) and we will compare our algorithm against the Lifelong learning approach of Chen et al. (2018) on a different benchmark dataset that they have introduced and provided publicly for researchers to experiment with.

In Wu et al. (2017), they have developed a multi-domain cross domain adaptation algorithm. However, they did not provide their exact test sets but they provided how they sampled their test reviews from the same benchmark Blitzer et al. (2007) dataset that we have used. Each of the four mentioned domains (Books, Kitchen, Electronics, DVD) contained 1000 positive and 1000 negative reviews. They have randomly sampled 500 positives and 500 negative reviews from each domain to form a 1000 sample test set for each domain. We have randomly sampled 500 positives and 500 negative samples in the same fashion then repeated this process 4 times until we had 5 randomly sampled test sets for each of these 4 domains. The accuracies and f1scores we have recorded for each domain on these 5 test sets were averaged to have a fairer comparison with Wu et al. (2017). Table 4-c shows the best average accuracy across the five test sets for each domain recorded.

Table 4-c Best average accuracies on the randomly generated test sets by ATLAS when using Euclidean distance

Average best accuracies on the randomly generated test sets				
Domain	Books	Kitchen	Electronics	DVD
Average Best Accuracy	0.7918	0.8762	0.8618	0.8082
Distance Measure	Euclidean	Euclidean	Euclidean	Euclidean

Table 4-d shows the best average accuracies recorded on the random test sets sampled for each of the four target domains when applying ATLAS using the Cosine distance similarity measure. Which is not significantly different from the results we recorded in Table 4-c.

Table 4-d: Best average accuracies recorded for ATLAS when using Cosine distance

Average best accuracies on the randomly generated test sets				
Domain	Books	Kitchen	Electronics	DVD
Average Best Accuracy	0.7918	0.8762	0.8626	0.8082
Distance Measure	Cosine	Cosine	Cosine	Cosine

For each of these 5 test sets, we have varied the classification threshold from 0.5 to 0.9 to record the effect of increasing precision on the overall performance. The best accuracy and

its corresponding threshold were recorded. The best accuracies achieved on the 5 datasets were averaged and recorded in the tables above. The best accuracies achieved across the 4 same domains in Wu et al. (2017) (ASDA) are shown in table 4-e. The authors of Wu et al. (2017) have compared their ASDA algorithm against approaches like SFA, SCL and others techniques mentioned in their paper. The best accuracy achieved by the ATLAS algorithm outperformed the accuracies reported in Wu et al. (2017) of the ASDA algorithm in each of the four target domains as shown in tables 4-c, 4-d, and 4-e.

Table 4-e: The best accuracies reported by Wu et al. (2017) (ASDA)

Best Accuracies ASDA algorithm				
Domain	Kitchen	Books	Electronics	DVD
Accuracy	0.8329	0.7508	0.8014	0.7764

The complexity of our approach could be analyzed by breaking down our ATLAS algorithm into five different parts that could be executed in series. N in the following steps represents the number of reviews per training set. Nf represents the number of unpacked features (N-gram counts, TFIDF values of ngrams). Number_Iterations represents the max number of iterations set for the classifier to converge. Batch_size represents the number of samples used for updating a single weight during training using stochastic gradient descent. The Batch_size could be set between 2 and the size of the training set. If we have k domains, the k-1 represents the number of source domains. Number_keys represent the number of words covered in the term frequency representation in the source and target dictionary/hash table. Finally, N_testsamples represents the number of test samples in the target domain at test time.

1. Training a single source classifier:
    - Time: O(Number_Iterations * (N * Nf + N + 1000*NF))

- Space: O(Number_Iterations * (N * Nf + N + batch_size*NF)) or O(Number_Iterations * (1 * Nf + 1 + batch_size *NF)) in case of loading the reviews one by one to memory.

2. Measuring distances from source to target domains.
    - Time: O(Number_keys) * (k-1)
    - Space: O(2*Number_keys) * k-1
3. Normalizing the distances.
    - Time: O(k-1)
    - Space: O(k-1)
4. Using the k-1 classifiers for predictions.
    - Time: O(k-1*N_testsamples)
    - Space: O(N_testsamples)
5. Aggregating the predictions in step 4
    - Time: (k-1)
    - Space: O(k-1)

We will dive into the performance of each of the three ensemble groups that we have mentioned earlier and highlight how each ensemble performed when compared against each ensemble's individual classifiers on the 5 randomly generated test sets in the following in-depth analysis section. We will compare our best result for each domain with the results published in Wu et al. (2017) in the in-depth analysis section and with the Liu et al. (2015) in the Bing Liu dataset section.

IN-DEPTH ANALYSIS ON THE BLITZER DATA SET

We will begin our analysis by showcasing the individual accuracies of the least performing and top performing classifier groups. As we have mentioned earlier, the top performing and least performing classifiers were identified by their domains' distances to the target domain. The top-performing classifiers are those classifiers trained on the closest n/2 domains to the target domain. While the least performing classifiers are those classifiers trained on the furthest n/2 domains to the target domain. We have mainly experimented with the Cosine and Euclidean distances with the term frequency representation where each domain was represented by a vector of adverb, verb, adjective and noun word counts that had a WordNet (Fellbaum et al. (1998)) sentiment score greater than 0.8. We have repeated our experiments twice on the Chen et al. (2018) dataset using the same representation for one set of experiments then using the same representation for the second set of experiments without using the WordNet sentiment scorer. For each domain, we will be showing an analysis of the average accuracies achieved by each individual classifier of the least performing and top performing classifiers groups when tested on the five randomly generated target domain test sets that were generated according to Wu et al. (2017). Each classifier is named after the domain that it was trained on. The Target Acc. field represents the accuracy of each of these classifiers on the target domain. The Cosine_distance and Euclidean_distance fields represent the un-normalized cosine and Euclidean distances. The similarity_rank field is the ascending order of the classifiers in terms of their domains' distances from the target domain. As we assume that as the distance between a source domain to a target domain decreases, the better should the average accuracy of the classifier be when tested on the five randomly generated test sets. The acc_rank is the actual order

of the classifiers' average accuracy on the target test sets from the best average accuracy to the worst such that a classifier that has an acc_rank equal to one is a classifier that has the best accuracy. While a classifier that has an acc_rank equal to 6, is a classifier that has the worst average accuracy on the generated test sets. The rank_diff field represents the unsigned difference between the acc_rank and the similarity_rank of each classifier. The lower this value for each classifier, the better the distance metric is in capturing a classifier's performance on the test set given its training domain's similarity to the target domain. As the similarity rank gets closer to the acc_rank for all classifiers, the summation of all the rank_diff values for all classifiers should be close to or equal to zero. As the summation of the rank_diff field values gets close to zero, the more the similarity measure is successful in capturing how these classifiers will perform on the target domain. We will start analyzing the results for each of the four target domains (The Book reviews, Kitchen reviews, Electronics reviews, and DVD reviews). We will start with the book reviews.

Table 4-f shows the accuracies on the book product reviews for each of the classifiers in the least performing group when using the Cosine distance as a way to rank them and separate them from the top performing classifiers and Table 4-g shows the individual accuracies of the least performing classifiers when using the Euclidean distance as the distance metric to separate them. We can see that the similarity_rank in table 4-f is identical to the rank of these classifiers by their average accuracy on the five randomly generated test sets (acc_rank). The size of the unlabeled book reviews used was 900,000+ according to table 3-a.

Table 4-f: Accuracies of the least performing classifiers (ranked by their cosine distance to the target) on the target domain

Books (least performing accs)					
Classifier	**Target Acc.**	**Cosine_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
baby	0.5856	0.677	1	1	0
beauty	0.5012	0.69	2	2	0
grocery	0.5008	0.719	3	3	0
gourmet	0.5	0.723	4	4	0
jewelrywatches	0.5	0.749	5	5	0
automotive	0.5	0.774	6	6	0

Similar to table 4-f above, the similarity_rank in table 4-g is identical to the average accuracy order of these classifiers on the target domain (Books reviews). Therefore, the rank difference is zero across all classifiers.

Table 4-g: Accuracies of the least performing classifiers (ranked by their Euclidean distance to the target) on the target domain

Books (least performing accs)					
Classifier	**Target Acc.**	**Euclidean_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
baby	0.5856	228.1	1	1	0
beauty	0.5012	229.13	2	2	0
grocery	0.5008	231.27	3	3	0
gourmet	0.5	231.57	4	4	0
jewelrywatches	0.5	233.25	5	5	0
automotive	0.5	234.77	6	6	0

Table 4-h shows the similarity_rank of the top performing classifiers when using the cosine distance as the similarity measure. Here, the similarity_rank is not identical to the acc_rank. The rank_diff for the electronics classifier is 4. Which means that its order in terms of its closeness to the target domain is 4 levels far from its order in terms of its actual average accuracy on the target test sets. The sum of the rank_diff across all the classifiers is equal to 7.

Table 4-h: Accuracies of the top performing classifiers (ranked by their cosine distance to the target) on the target domain

Books (top performing accs)					
Classifier	**Target Acc.**	**Cosine_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
dvd	0.7928	0.239	1	1	0
electronics	0.6832	0.547	6	2	4
kitchen	0.7032	0.554	4	3	1
healthpersonal	0.7208	0.624	2	4	2
cameraphoto	0.688	0.642	5	5	0
apparel	0.7172	0.627	3	6	3

Similarly, table 4-i shows that the similarity_rank of the top performing classifiers is not the same as the acc_rank. The total rank difference is equal to 10. As the total rank difference moves away from zero, the less information is captured by the Euclidean Distances from the source to the target domains about the performance of the classifiers on the target domain. However, it is important to note that the maximum average accuracy recorded in the least performing classifiers in tables 4-f and 4-g is less than the minimum average accuracy recorded for one of the top performing classifiers. Which shows that the Euclidean and Cosine distances were successful in separating the n/2 classifiers that had high average accuracies on the test set from the n/2 classifiers that had the worst average accuracies on the average test sets without having any knowledge of their actual average test accuracies. Where n is the number of the source domain classifiers. If a classifier in the least performing group has a better accuracy than a classifier in the top performing group, we call that a “miss”. We will show later the number of misses when separating the top performing from the least performing classifiers.

Table 4-i: Accuracies of the top performing classifiers (ranked by their Euclidean distance to the target) on the target domain

Books (top performing accs)					
Classifier	**Target Acc.**	**Euclidean_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
dvd	0.7928	156.7	1	1	0
electronics	0.6832	214.91	6	2	4
kitchen	0.7032	215.76	4	3	1
healthpersonal	0.7208	223.29	2	4	2
cameraphoto	0.688	225.03	5	5	0
apparel	0.7172	225.27	3	6	3

Table 4-j sums up the sum of the rank_diff field for the least and top performing groups across the cosine and Euclidean distance formed groups. It is clear that for the top performing group, the rank_diff total was far from zero when using either distance measure. Which shows that the cosine distance metric when utilized on the Blitzer et al. (2007) dataset where domains have different training set sizes, leads to a better similarity measure in terms of predicting how the source classifiers would perform on the target domain than the Euclidean distance. With that being said, both distance metrics were successful in grouping the least performing together and the top performing classifiers together as in Table 4-k, we see zero "misses". "misses" are the number of classifiers in the top performing group that have a lower average accuracy on the target domain than the top average accuracy recorded for the lowest performing group.

Table 4-j: The sum of the rank_diff for each of the four groups

Distance Metric	**Classifiers group**	**Target domain**	**Sum_rank_diff**
Cosine	Least Performing	Books	0
Cosine	Top Performing	Books	10
Euclidean	Least Performing	Books	0
Euclidean	Top Performing	Books	10

Table 4-k: Number of misses when using each distance measure in the ATLAS algorithm

Distance Metric	**Domain**	**Misses**
Cosine	Books	0
Euclidean	Books	0

After varying the threshold values from 0.5 to 0.9. We have noticed that the best average accuracies and average F1_Scores were achieved when setting the threshold values between 0.5 and 0.7. Table 4-l shows the average accuracy and average f1_scores recorded for the three ensemble groups when having threshold values between 0.5 and 0.7. The best average accuracy and average f1score recorded were 0.7918 and 0.7994 respectively when combining the top performing classifiers at a threshold of 0.5. These values were better than the accuracy and f1scores reported by Wu et al. (2017) algorithm (ASDA) which were 0.7508 and 0.7501 respectively.

Table 4-l: Average accuracy, and average F1Score recorded while varying the ensemble group and the classification threshold

Books					
Group	Threshold	Average Accuracy (Euclidean)	Average Fscore (Euclidean)	Average Accuracy (Cosine)	Average Fscore (Cosine)
Least Performing	0.5	0.5012	0.6672	0.5012	0.6672
Top Performing	0.5	0.7918	0.7994	0.7918	0.7994
Top and Least	0.5	0.7276	0.7797	0.7276	0.7797
Least Performing	0.6	0.502	0.6675	0.502	0.6675
Top Performing	0.6	0.7724	0.7623	0.7724	0.7623
Top and Least	0.6	0.7764	0.796	0.7854	0.8117
Least Performing	0.7	0.5856	0.6978	0.5856	0.6978
Top Performing	0.7	0.7458	0.6892	0.7494	0.7016
Top and Least	0.7	0.7758	0.7638	0.7758	0.7638

We will move to the Kitchen product reviews domain to check on how ATLAS performs. Similar to the books product reviews, we have tested the least performing classifiers and the top performing individual classifiers on the five randomly generated test sets then averaged their accuracies and recorded them in the following four tables. Table 4-m shows that the acc_rank is not similar to the similarity_rank which is okay as the similarity_rank should not coincide with the acc_rank. However, we have noticed that the summation of

the rank_diff when using the cosine distance metric is equal to 10. While the summation of the rank_diff when using the Euclidean distance of the least performing classifiers is equal to 16 which means that the cosine distance metric captured more information about the classifiers' performance on the test set when compared against the Euclidean distance metric. As we have mentioned earlier, as the summation of the rank_diff becomes close to zero, the better is the distance metric in being able to capture how will the classifiers will function on the test sets as it means that the similarity rank is closer to the acc_rank of the classifiers. However, both distance measures led the ATLAS to attain similar average accuracies and F1Scores when combining the source classifiers as shown in table 4-s.

Table 4-m: Accuracies of the least performing classifiers (ranked by their cosine distance to the target) on the target domain (Kitchen)

Kitchen (least performing accs)					
Classifier	**Target Acc.**	**Cosine_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
grocery	0.5018	0.487	3	1	2
dvd	0.742	0.492	1	2	1
gourmet	0.5	0.503	6	3	3
jewelrywatches	0.501	0.524	4	4	0
book	0.6604	0.554	2	5	3
automotive	0.5006	0.563	5	6	1

Table 4-n: Accuracies of the least performing classifiers (ranked by their Euclidean distance to the target) on the target domain (Kitchen)

Kitchen (least performing accs)					
Classifier	**Target Acc.**	**Euclidean_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
gourmet	0.5	96.97	6	1	5
computervideo	0.5028	97.01	3	2	1
jewelrywatches	0.501	97.77	4	3	1
automotive	0.5006	99.84	5	4	1
dvd	0.742	167.77	1	5	4
book	0.6604	215.76	2	6	4

Similar to tables 4-m and 4-n, Tables 4-o and 4-p show the average accuracies of the top performing classifiers on the kitchen test sets. The sum of the rank_diff field is equal to 6 as summarized in Table 4-q. While it is equal to 12 when using the Euclidean distance as

shown in table 4-p and 4-q. We could see in table 4-r that we have one miss. Which means that the classifier that has the maximum average accuracy in the least performing group has a greater average accuracy than one classifier in the top performing group. In that case, it's the beauty domain classifier. That miss applies for the groups separated by the Euclidean and Cosine distances.

Table 4-o: Accuracies of the top performing classifiers (ranked by their cosine distance to the target) on the target domain(kitchen)

Kitchen (top performing accs)					
Classifier	**Target Acc.**	**Cosine_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
electronics	0.8244	0.36	2	1	1
healthpersonal	0.8216	0.368	3	2	1
cameraphoto	0.812	0.407	4	3	1
apparel	0.8326	0.408	1	4	3
baby	0.805	0.411	5	5	0
beauty	0.5004	0.461	6	6	0

Table 4-p: Accuracies of the top performing classifiers (ranked by their Euclidean distance to the target) on the target domain (Kitchen)

Kitchen (top performing accs)					
Classifier	**Target Acc.**	**Euclidean_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
healthpersonal	0.8216	88.91	3	1	2
baby	0.805	90.62	5	2	3
cameraphoto	0.812	91.81	4	3	1
apparel	0.8326	91.98	1	4	3
electronics	0.8244	94.66	2	5	3
beauty	0.5004	94.75	6	6	0

According to table 4-q, it is clear from our observations that the cosine distance proved to be a better ranker of the least performing and top performing classifiers than the Euclidean distance. As the sum_rank_diff for the least performing and top performing classifiers that were ranked by their cosine distance to the target domain was smaller than the sum_rank_diff of the least and top classifiers ranked by their Euclidean distances to the target domain.

Table 4-q: The sum of the rank_diff for each of the four groups (tested on Kitchen reviews)

Distance Metric	Classifiers group	Target domain	Sum_rank_diff
Cosine	Least Performing	Kitchen	10
Cosine	Top Performing	Kitchen	6
Euclidean	Least Performing	Kitchen	16
Euclidean	Top Performing	Kitchen	12

We will explain later in the chapter why the cosine distance metric was better at ranking the least performing and top performing classifiers such that they are ranked closer to the acc_rank than the Euclidean distance. It is important to note, that the purpose of measuring the distances from the source to target domains was to calculate classifier weights that would then be used in boosting the three ensembles that we have grouped and not to necessarily rank the least and top performing classifiers. The rank_diff and sum_rank_diff fields are used to get a sense of which distance measure when used on top of the term frequency representation mentioned earlier, leads to a classifier ranking that captures their average accuracy on the test set order. The number of "misses" is for identifying the number of top performing classifiers that have an average accuracy that is lower than the max average accuracy recorded in the least performing classifiers group and by that we would know if the n/2 least performing classifiers were adequately separated from the n/2 top performing classifiers given that we have n source classifiers. In our case n is equal to 13. As we have mentioned earlier, table 4-r shows the number of classifiers in the top performing group that had an average accuracy that is smaller than the maximum average accuracy recorded in the least performing group when using the cosine and Euclidean distances.

Table 4-r: Number of misses when using each distance measure in the ATLAS algorithm

Distance Metric	Domain	Misses
Cosine	Kitchen	1
Euclidean	Kitchen	1

Table 4-s shows the average accuracy and Average F1_scores of the ATLAS algorithm when used in aggregating the outputs of three different groups of classifiers (Least performing, top performing, and the top and least performing) while varying the classification thresholds from 0.5 to 0.7. The maximum average accuracy recorded was 0.8762 when using ATLAS on top of the top performing classifiers with a threshold of 0.6. While the max F1_score was achieved at a threshold of 0.6 when using the top and least performing classifiers with the ATLAS (0.8788). The max-average accuracy and max average F1_Scores were greater than the accuracy and f1score achieved by Wu et al. (2017) (ASDA algorithm) which were 0.8329 and 0.8328 respectively.

Table 4-s: Average accuracy, and average F1Score recorded while varying the ensemble group and the classification threshold on the kitchen domain

Kitchen					
Group	Threshold	Average Accuracy (Euclidean)	Average Fscore (Euclidean)	Average Accuracy (Cosine)	Average Fscore (Cosine)
Least Performing	0.5	0.5038	0.6683	0.502	0.6675
Top Performing	0.5	0.8678	0.8719	0.8748	0.8767
Top and Least	0.5	0.8204	0.8423	0.8386	0.8537
Least Performing	0.6	0.751	0.7541	0.7514	0.7544
Top Performing	0.6	0.8762	0.8775	0.8762	0.8775
Top and Least	0.6	0.8734	0.8788	0.8698	0.8657
Least Performing	0.7	0.7426	0.7259	0.6524	0.5041
Top Performing	0.7	0.8554	0.8406	0.8554	0.8406
Top and Least	0.7	0.8596	0.8474	0.8396	0.8186

Moving on to the Electronics product reviews domain. Similar to the Kitchen and book product reviews, the following tables show the accuracies on the five randomly generated target domain (electronics) test sets. As mentioned earlier, these test sets were sampled in a similar fashion as in Wu et al. (2017). Tables 4-t and 4-u show the individual average accuracies of the least performing classifiers when using the Cosine and Euclidean

distances. We can see that we have one miss here. Where the book classifier that is part of the least performing classifier groups had a better average accuracy than the computer video classifier in the top performing groups when using either the Euclidean or Cosine distances for separating the classifiers.

Table 4-t: Accuracies of the least performing classifiers (ranked by their cosine distance to the target) on the target domain (Electronics)

Electronics (least performing accs)					
Classifier	**Target Acc.**	**Cosine_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
beauty	0.5036	0.5	2	1	1
jewelrywatches	0.502	0.51	4	2	2
grocery	0.5036	0.54	3	3	0
book	0.6466	0.54	1	4	3
automotive	0.5004	0.55	5	5	0
gourmet	0.5	0.56	6	6	0

Table 4-u: Accuracies of the least performing classifiers (ranked by their Euclidean distance to the target) on the target domain (Electronics)

Electronics (least performing accs)					
Classifier	**Target Acc.**	**Euclidean_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
beauty	0.5036	99.4	4	1	3
automotive	0.5004	100.4	5	2	3
grocery	0.5036	101.4	3	3	0
gourmet	0.5	102.3	6	4	2
dvd	0.7218	165.7	1	5	4
book	0.6466	214.9	2	6	4

The similarity_rank is not identical to the acc_rank of the least performing classifiers which is okay. The sum of the rank_diff in table 4-t is smaller than the sum of the rank_diff of the least performing classifiers ranked by the Euclidean distances as shown in table 4-u. On the other hand, tables 4-v and 4-w show the individual average accuracies of these classifiers on the target test sets (Electronics test sets). Also, when using the cosine distance in ranking these top-performing classifiers, the similarity rank is identical to the acc_rank as shown in table 4-v. Which again, brings up the observation

that the cosine distance measure shows to be better at ranking the least performing and top performing classifiers in a way that makes their similarity_rank closer to their actual acc_rank on the target domain's test sets.

Table 4-v: Accuracies of the top performing classifiers (ranked by their cosine distance to the target) on the target domain(Electronics)

Electronics (top performing accs)					
Classifier	**Target Acc.**	**Cosine_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
cameraphoto	0.8154	0.35	1	1	0
kitchen	0.8012	0.36	2	2	0
healthpersonal	0.794	0.4	3	3	0
apparel	0.788	0.41	4	4	0
baby	0.764	0.42	5	5	0
computervideo	0.5052	0.43	6	6	0

Table 4-w shows the top performing classifiers ranked by their domains' similarity to the target domain's reviews using the Euclidean distance as the similarity measure. We could see that unlike in Table 4-v, the similarity_rank is not identical to the acc_rank.

Table 4-w: Accuracies of the top performing classifiers (ranked by their Euclidean distance to the target) on the target domain(Electronics)

Electronics (top performing accs)					
Classifier	**Target Acc.**	**Euclidean_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
cameraphoto	0.8154	87.4	1	1	0
baby	0.764	93.3	5	2	3
healthpersonal	0.794	93.7	3	3	0
apparel	0.788	93.79	4	4	0
computervideo	0.5052	94.55	6	5	1
kitchen	0.8012	94.66	2	6	4

Table 4-x, shows again that the cosine distance similarity metric is better at ranking the top and least performing classifiers in a way such that their similarity_rank is closer to their acc_rank on the target test set compared with the Euclidean distance measure. We will discuss why that observation is recurring at the end of our Blitzer results.

Table 4-x: The sum of the rank_diff for each of the four groups (tested on Electronics reviews)

Distance Metric	Classifiers group	Target domain	Sum_rank_diff
Cosine	Least Performing	Electronics	6
Cosine	Top Performing	Electronics	0
Euclidean	Least Performing	Electronics	16
Euclidean	Top Performing	Electronics	8

Table 4-y shows the number of "misses" that we have alluded to earlier that represents the number of top performing classifiers that had an average accuracy smaller than the maximum accuracy recorded for the least performing classifiers when using either distance metric.

Table 4-y: Number of misses when using each of the following distance measures in the ATLAS algorithm

Distance Metric	Domain	Misses
Cosine	Electronics	1
Euclidean	Electronics	1

Table 4-z shows the average accuracies and average f1scores achieved when varying the threshold from 0.5 to 0.7. The best average accuracy recorded was 0.8626 when applying the ATLAS algorithm using the top performing classifiers and a threshold of 0.6 which is better than any of the individual accuracies recorded for the top performing and least performing classifiers in addition to the best accuracy recorded in Wu et al. (2017) that was equal to 0.8014. Which indicates that the ATLAS algorithm was able to boost the performance of these individual classifiers. The best average F1-Score recorded was 0.8587 which is higher than the best F1Score reported in Wu et al. (2018) which was 0.8011.

Table 4-z: Average accuracy, and average F1Score recorded while varying the ensemble group and the classification threshold on the Electronics domain

Electronics					
Group	Threshold	Average Accuracy (Euclidean)	Average Fscore (Euclidean)	Average Accuracy (Cosine)	Average Fscore (Cosine)
Least Performing	0.5	0.5058	0.6692	0.5032	0.668
Top Performing	0.5	0.8618	0.8633	0.8612	0.8564
Top and Least	0.5	0.8282	0.8474	0.822	0.8404
Least Performing	0.6	0.7382	0.7317	0.5044	0.6686
Top Performing	0.6	0.8598	0.8538	0.8598	0.8538
Top and Least	0.6	0.859	0.8587	0.8626	0.8584
Least Performing	0.7	0.7222	0.6935	0.6492	0.5167
Top Performing	0.7	0.8122	0.777	0.8122	0.777
Top and Least	0.7	0.8182	0.7864	0.8306	0.805

Finally, switching to the DVD domain. Similar to the previous target domains. The least performing source domain classifiers were ranked based on their domains' similarity to the DVD domain in tables 4-aa and 4-bb using the Cosine and Euclidean distances. Here, both distances had the same sum of rank_diff while having zero misses.

Table 4-aa: Accuracies of the least performing classifiers (ranked by their cosine distance to the target) on the target domain(DVD)

DVD (least performing accs)					
Classifier	**Target Acc.**	**Cosine_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
baby	0.651	0.614	1	1	0
beauty	0.502	0.634	2	2	0
grocery	0.501	0.668	3	3	0
gourmet	0.5	0.673	5	4	1
jewelrywatches	0.5002	0.697	4	5	1
automotive	0.5	0.728	6	6	0

Table 4-bb: Accuracies of the least performing classifiers (ranked by their Euclidean distance to the target) on the target domain(DVD)

DVD (least performing accs)					
Classifier	**Target Acc.**	**euclidean_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
baby	0.651	179.25	1	1	0
beauty	0.502	180.8	2	2	0
grocery	0.501	183.26	3	3	0
gourmet	0.5	183.56	5	4	1
jewelrywatches	0.5002	185.06	4	5	1
automotive	0.5	186.8	6	6	0

Table 4-cc: Accuracies of the top performing classifiers (ranked by their cosine distance to the target) on the target domain(DVD)

DVD (top performing accs)					
Classifier	**Target Acc.**	**Cosine_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
book	0.7268	0.239	3	1	2
electronics	0.7186	0.476	4	2	2
kitchen	0.7002	0.492	6	3	3
healthpersonal	0.738	0.562	1	4	3
cameraphoto	0.7082	0.574	5	5	0
apparel	0.7374	0.577	2	6	4

Table 4-dd: Accuracies of the top performing classifiers (ranked by their Euclidean distance to the target) on the target domain(DVD)

DVD (top performing accs)					
Classifier	**Target Acc.**	**euclidean_distance**	**acc_rank**	**optimal_rank**	**rank_diff**
book	0.7268	156.7	3	1	2
electronics	0.7186	165.71	4	2	2
kitchen	0.7002	167.77	6	3	3
healthpersonal	0.738	174.79	1	4	3
cameraphoto	0.7082	175.77	5	5	0
apparel	0.7374	176.07	2	6	4

Table 4-ee shows the summation of the rank_diff field when using the cosine and Euclidean distances in ranking the least and top performing classifiers.

Table 4-ee: The sum of the rank_diff for each of the four groups (tested on DVD reviews)

Distance Metric	**Classifiers group**	**Target domain**	**Sum_rank_diff**
Cosine	Least Performing	DVD	2
Cosine	Top Performing	DVD	14
Euclidean	Least Performing	DVD	2
Euclidean	Top Performing	DVD	14

Again, the cosine distance based ATLAS resulted in having a smaller difference between the similarity_rank and the acc_rank of the least and top performing classifiers when compared against the sum_rank_diff of the least and top performing groups ranked by Euclidean distance to the target.

According to Table 4-ff, the best average accuracy achieved by the ATLAS algorithm was 0.8082 when using the top performing classifier group at a threshold of 0.5. While the best

average F1score recorded was 0.8034 when the top and least performing classifiers were combined using the ATLAS algorithm. The average accuracy achieved by the ATLAS algorithm was better than the best accuracy achieved by the ASDA algorithm (Wu. et al. (2017)) which was 0.7764. Also, the best accuracy recorded was better than the individual average accuracies recorded for the least and top performing classifiers. Which indicates that the ATLAS boosted the classifiers' individual average accuracies on the target domain's test sets. The best average F1Score on the target domain recorded was 0.8034 which is higher than the F1Score recorded for ASDA which was 0.7759.

Table 4-ff: Average accuracy, and average F1Score recorded while varying the ensemble group and the classification threshold on the DVD domain

DVD					
Group	Threshold	Average Accuracy (Euclidean)	Average Fscore (Euclidean)	Average Accuracy (Cosine)	Average Fscore (Cosine)
Least Performing	0.5	0.5028	0.6679	0.5028	0.6679
Top Performing	0.5	0.8082	0.7947	0.8082	0.7947
Top and Least	0.5	0.7332	0.7764	0.7332	0.7764
Least Performing	0.6	0.5028	0.6679	0.5028	0.6679
Top Performing	0.6	0.8022	0.7833	0.8022	0.7833
Top and Least	0.6	0.7982	0.8034	0.7982	0.8034
Least Performing	0.7	0.651	0.7302	0.651	0.7302
Top Performing	0.7	0.7418	0.6673	0.7418	0.6673
Top and Least	0.7	0.8002	0.7778	0.8002	0.7778

COSINE VS EUCLIDEAN DISTANCES

When experimenting with the cosine and Euclidean distances, both resulted in weights that helped the ATLAS algorithm in achieving high average accuracies on the target domains' test sets. The difference between the best average accuracy and the best average F1Scores when using either the Euclidean or Cosine distances were insignificant. However, when it came to ranking the classifiers by their domains' similarity to the target domain in an order that is similar to their average accuracy ranking on the target domain's test sets then the cosine distance measure had an edge over the Euclidean distance measure as we have shown in our results on the four target domains earlier. We believe the reason behind that is that the cosine distance measure normalizes the term frequency values in a way such that it converts the term frequency representation to a representation that is similar to the term frequency-inverse document frequency representation mentioned in the previous chapter. Therefore, making the cosine distance metric invariant of the lengths of the two dictionaries being compared at any point in time. On the other hand, the Euclidean distance metric does not normalize the term frequencies stored in the source and target domain dictionaries, therefore it is not invariant of the lengths of the two input dictionaries and features/terms that have high frequencies and occur throughout both dictionaries (source and target dictionaries) do not get penalized by normalization. Nonetheless, the Euclidean distance measure had the same number of "misses" as the cosine distance metric. Which indicates that it can effectively separate the least performing from the top performing classifiers in the ensembles similar to the cosine distance metric.

OTHER REPRESENTATION TECHNIQUES

Before we conclude with our results on the Blitzer et al. (2007), we would like to mention that we have tried two different representation techniques for the source and target domains. We have tried representing the source and target domains by training 2 word2vec and 2 doc2vec models on all of the reviews included in all of the 14 domains. Since the book, product reviews domain has over 900,000 reviews and the second largest domain is the DVD domain that has 145,000 reviews and all the other domains have to review sizes in the 10s of thousands. So we have capped the number of reviews per domain to a max of 25,000 reviews for our embedding models do not get skewed by a large number of reviews in the books domain. The two word2vec models were trained to produce word2vec vectors of sizes 200 and 400 respectively. Similarly, the doc2vec models were trained to produce vectors of sizes of 200 and 400. These models were then used in representing each domain. Consequently, when using the word2vec_200 dimensional model, the word2vec vector of each word in each review of a domain is calculated then the average of the word2vec vectors of all the words in the review is calculated and stored. Afterward, the mean of all the average word2vec vectors of all reviews is calculated and that is the vector that is used in representing the domain. We refer to it as the mean average word2vec vector of the domain. The same process is repeated with the word2vec_400 dimensional model to get the mean average 400-dimensional word2vec vectors of each domain. As for doc2vec, we have utilized the doc2vec_200 dimensional model for calculating the 200-dimensional doc2vec vector of each review in each domain. Later on, the average of all the 200-dimensional doc2vec vectors in each domain was calculated. The process is repeated with

the 400-dimensional doc2vec model and we ended up with four representations for each of the 14 domains that we have experimented with so far.

DVD

Table 4-gg: Max accuracy-200 d2v vector rep.

Doc2Vec200 - DVD			
Threshold	Group	distance	Max Acc
0.7	Top_Least	Euc	0.812
0.7	Top_Least	Cosine	0.813

Table 4-hh: Max accuracy-400 d2v vector rep.

Doc2Vec400 - DVD			
Threshold	Group	distance	Max Acc
0.7	Top_Least	Euc	0.75
0.7	Top_Least	Cosine	0.755

Table 4-ii: Max accuracy-200 w2v vector rep.

Word2Vec200 - DVD			
Threshold	Group	distance	Max Acc
0.7	Top_Least	Euc	0.7675
0.7	Top_Least	Cosine	0.755

Table 4-jj: Max accuracy-400 w2v vector rep.

Word2Vec400 - DVD			
Threshold	Group	distance	Max Acc
0.8	Top_Least	Euc	0.765
0.8	Top_Least	Cosine	0.755

Kitchen

Table 4-kk: Max accuracy-200 d2v vector rep.

Doc2Vec200 - Kitchen			
Threshold	Group	distance	Max Acc
0.6	Top_Least	Euc	0.845
0.7	Top_Least	Cosine	0.8475

Table 4-ll: Max accuracy-400 d2v vector rep.

Doc2Vec400 - Kitchen			
Threshold	Group	distance	Max Acc
0.6	Top_Least	Euc	0.845
0.7	Top_Least	Cosine	0.8475

Table 4-mm: Max accuracy-200 w2v vector rep.

Word2Vec200 - Kitchen			
Threshold	Group	distance	Max Acc
0.6	Top	Euc	0.858
0.6	Top_Least	Cosine	0.846

Table 4-nn: Max accuracy-400 w2v vector rep.

Word2Vec400 - Kitchen			
Threshold	Group	distance	Max Acc
0.6	Top	Euc	0.835
0.5	Top_Least	Cosine	0.8625

Books

Table 4-oo: Max accuracy-200 d2v vector rep.

Doc2Vec200 - Books			
Threshold	Group	distance	Max Acc
0.7	Top_Least	Euc	0.745
0.7	Least	Cosine	0.74

Table 4-pp: Max accuracy-400 d2v vector rep.

Doc2Vec400 - Books			
Threshold	Group	distance	Max Acc
0.7	Least	Euc	0.74
0.8	Least	Cosine	0.74

Table 4-qq: Max accuracy-200 w2v vector rep.

Word2Vec200 - Books			
Threshold	Group	distance	Max Acc
0.7	Top_Least	Euc	0.7325
0.7	Least	Cosine	0.7025

Table 4-rr: Max accuracy-400 w2v vector rep.

Word2Vec400 - Books			
Threshold	Group	distance	Max Acc
0.6	Top_Least	Euc	0.7325
0.7	Top_Least	Cosine	0.735

Electronics

Table 4-ss: : Max accuracy-200 d2v vector rep.

Doc2Vec200 - Electronics			
Threshold	Group	distance	Max Acc
0.6	Top_Least	Euc	0.82
0.6	Top_Least	Cosine	0.8275

Table 4-tt: Max accuracy-400 d2v vector rep.

Doc2Vec400 - Electronics			
Threshold	Group	distance	Max Acc
0.6	Top_Least	Euc	0.82
0.6	Top_Least	Cosine	0.8275

Table 4-uu: Max accuracy-200 w2v vector rep.

Word2Vec200 - Electronics			
Threshold	Group	distance	Max Acc
0.5	Top	Euc	0.83
0.7	Top	Cosine	0.82

Table 4-vv: Max accuracy-400 w2v vector rep.

Word2Vec400 - Electronics			
Threshold	Group	distance	Max Acc
0.5	Top	Euc	0.83
0.6	Top_Least	Cosine	0.7925

As shown in tables 4-gg to 4-xx, the ATLAS results collected with these representations were not as good as the ATLAS results recorded when representing the domains as a dictionary of the frequency of their verbs, adverbs, adjectives, and nouns as we have shown in previous sections when testing on the four main target domains.

CONCLUDING BLITZER RESULTS

The results achieved so far using the ATLAS algorithm on the Blitzer et al. (2007) have proved to be better than the results achieved by the ASDA algorithm which were published back in Wu et al. (2017) paper published in the ACL 2017 conference. These results stress on the value added by our algorithm when compared against the latest multi-source domain transfer learning system published in ACL. In the following section, we will show our results on the Chen et al. (2018) benchmark dataset and compare our results with the results published in the same paper. In our previous experiments on the Blitzer et al. (2007) dataset, we have used the WordNet sentiment scorer to pick the verb, adverbs, adjective and noun terms that had a WordNet positive or negative sentiment score greater than 0.8. When testing our ATLAS algorithm on the Bing Liu benchmark dataset (Chen et al. (2018), we wanted to test our algorithm when using the WordNet sentiment scorer and when not using it. In both cases, the ATLAS proved to deliver better results than the results published in Chen et al. (2018). The following figures show the average accuracy and average F1Score of the Wu et al. (2017) ASDA algorithm on the four test domains we covered versus the average accuracy and average F1Score of our approach (ATLAS) across the four test domains.

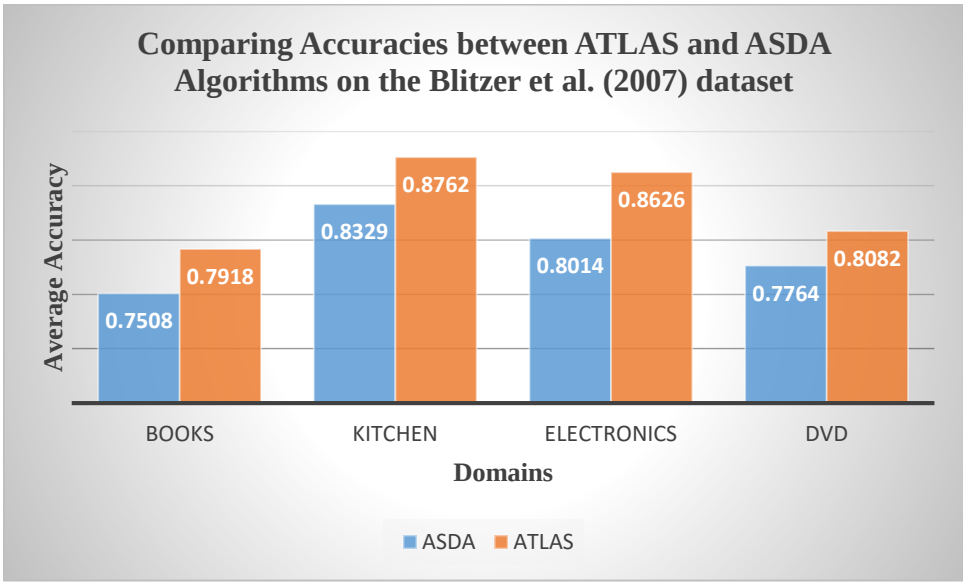

Figure 4-a: Best accuracies recorded when using ATLAS vs ASDA (Wu et al. (2017))

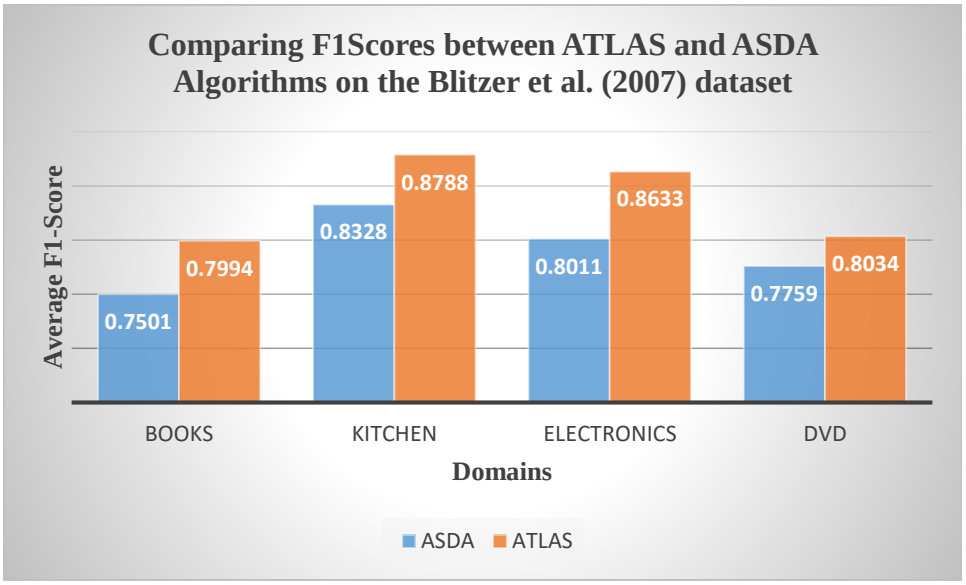

Figure 4-b: Best F1Scores recorded when using ATLAS vs ASDA (Wu et al. (2017))

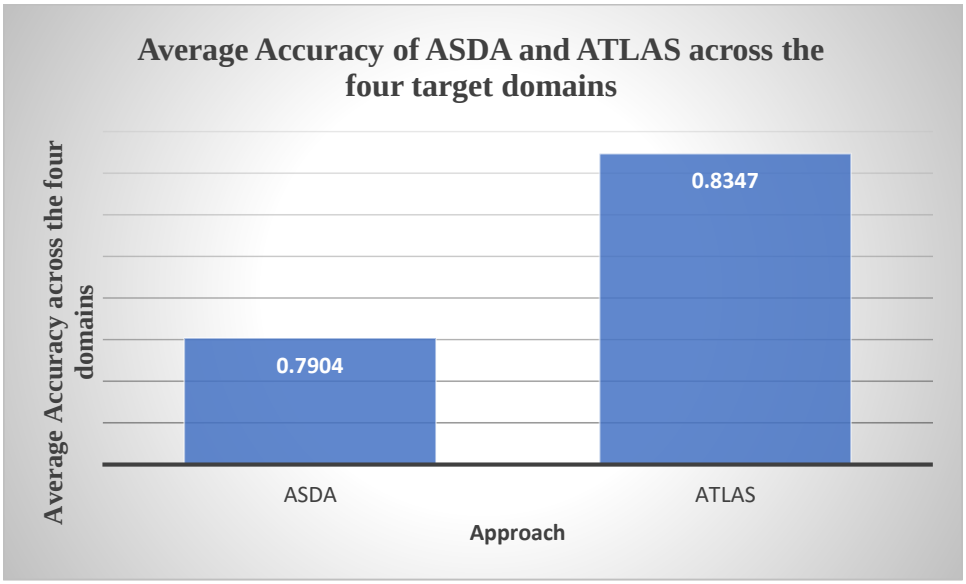

Figure 4-c: Average accuracy across the four target domains when using ATLAS vs ASDA (Wu et al. (2017))

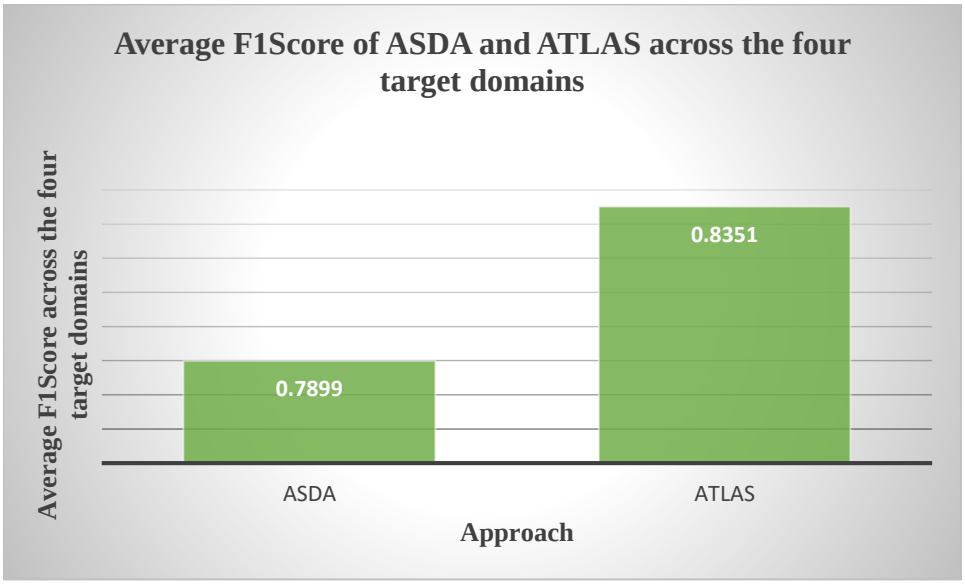

Figure 4-d: Average F1Score across the four target domains when using ATLAS vs ASDA (Wu et al. (2017))

Figures 4 through 7 highlights the improvement in accuracy and F1Score added by our ATLAS algorithm. Figure 4 shows when applying the transfer learning module of ATLAS on the four target domains in Wu et al. (2017), ATLAS proved to give a better accuracy and F1Score on all of these four domains. The accuracy shown in figure 4 for the ASDA algorithm was reported in Wu et al. (2017). These were the best accuracy recorded for the ASDA algorithm when applying it on a randomly sampled balanced test set comprised of 500 positive samples and 500 negative samples. We have sampled the balanced test set for each domain in the same way as in Wu et al. (2017) five times creating five 1000 review test sets for each of the four target domains. We have applied our ATLAS algorithm on the five test sets in each domain, then averaged the five accuracies for each domain to get an average accuracy on each domain. We have repeated the same process when calculating the average F1Score for each domain. Figures 4 and 5 compare the average accuracies and F1scores of the ATLAS algorithm against the best accuracy reported in Wu et al. (2017) for their ASDA approach. We wanted to sample five test sets and average the accuracies to better evaluate our model against the ASDA algorithm. Figures 6 shows the average of the four accuracies shown in Figure 4 for the ASDA and ATLAS while Figure 7 shows the average of the four F1Scores in Figure 5 for the ASDA and ATLAS which shows that on average, there is a 4-point improvement in accuracy and a 4-point improvement in F1score across the four domains.

APPLYING ATLAS ON THE BING LIU DATA SET

We have tested our ATLAS algorithm on a different benchmark dataset offered by Professor Bing Liu of the University of Illinois, Chicago that was used in Chen et al. (2018) paper published in ACL 2015 and an updated version of the paper published in Arxiv 2018. We will compare our results to the version published in January of 2018 to Arxiv. The dataset consists of twenty different domains. Each domain contains 1000 labeled reviews that were given either positive, negative or neutral labels. Chen et al. (2018) trained a Life Long learning system on the positive and negative labeled reviews of 19 domains and left one domain out for testing. They have repeated this process twenty times keeping one of the 20 domains as a target test set then, they have averaged their system's accuracies and f1scores across the twenty target test domains which we will compare against. Life Long learning is similar to Transfer Learning except that according to Chen et al. (2018), Life Long learning is a process that utilizes knowledge from k-1 domains then applies this knowledge on the kth domain without using any labeled or unlabeled data from the kth target domain during training. On the other hand, Transfer Learning system learns from k-1 domains in addition to learning from the unlabeled or labeled samples from the target domain. A transfer learning method that uses labeled and unlabeled samples from the k-1 source domains in addition to unlabeled samples only from the target domain is called an unsupervised transfer learning method. While a transfer learning method that uses labeled and unlabeled data from the k-1 source domains in addition to labeled and unlabeled samples from the target domain is called a semi-supervised transfer learning method or a semi-supervised cross domain adaptation method. Finally, a transfer learning approach that uses labeled and unlabeled data from the k-1 source domains and labeled samples only

from the target domain is called a supervised transfer learning approach or a supervised cross domain adaptation method. The ATLAS algorithm is an unsupervised transfer learning method that uses labeled and unlabeled reviews from the k-1 source domains in addition to unlabeled data from the target domain in measuring the distances from each source domain to the target domain. However, with the Chen et al. (2018) data set, we have added a slight transformation step that transforms our algorithm from a transfer learning approach to a lifelong learning approach as we will show shortly. First, we will mention how we measured the distances from the source domains to the target domains then will cover how the test sets were generated to match the same class distribution mentioned in Chen et al. (2018). Afterward, we will show how the ATLAS algorithm is transformed into a lifelong learning algorithm. The distance metrics we have used in measuring the distances from multiple source domains to the target domain. Since we only have 1000 reviews per domain in this dataset. We have tried measuring the distances by measuring the distance from the source domain's 1000 review datasets to the unlabeled test samples of the target domain. Since the target test set is sampled from the target domain's 1000 labeled samples, the unlabeled target domain training samples are not enough for calculating the distance from each source to target for calculating the classifier weights prior to test time. As a result, at test time, we have measured the distance of the full unlabeled test set in addition to some remaining unlabeled training samples to every source domain's training set without the use of any labels in the source or target domains. When transforming our ATLAS method from the transfer learning mode to the lifelong learning mode, we have measured the distance of each test review in the target domain's test set from the unlabeled 19 source samples at test time. So the weights of the source domain classifiers were

modified when applying ATLAS on each test sample. It is a complex operation but it did not require having an unbalanced training sample before testing the system on the test set. It did not require also having the full test set before applying the ATLAS system. As the distance was calculated from each source to the target one test sample at a time without requiring to have the full test set stored in advance. The domains were represented using the term frequency representation mentioned earlier, where the verbs, adverbs, adjectives, and nouns that had a sentiment score greater than 0.8 were saved in a dictionary with their counts. We have also repeated this representation without using the WordNet sentiment scorer at all and we will show the results delivered when using the WordNet sentiment scorer for filtering verbs, adverbs, adjectives, and nouns having a positive or negative sentiment score greater than 0.7 or not shortly. We noticed that there was no noticeable difference in the results achieved when using the WordNet sentiment scorer or not. The distance from each source to target domains was measured by the cosine and Euclidean distances between their corresponding dictionaries. In the transfer learning and lifelong learning sections of our results, all source domains were represented by their term frequency dictionaries. However, the target domain was represented as a term frequency dictionary of the verbs, adverbs, adjectives, and nouns that occurred in the target domain's test set in addition to some unlabeled target domain samples that were not included in the test set. On the other hand, for the lifelong learning section of our results the target domain's dictionary changes with every review in the test set, as the dictionary consists of the term frequencies of the verbs, adverbs, adjectives, and nouns that occurred in the target domain's individual test review that was being tested at test time. As the target domain's term frequency dictionary is created at test time for each test review at a time. The distance

between a source domain's dictionary and a target domain's dictionary was measured using the Euclidean and cosine distance functions mentioned earlier. We first calculated the distances using the WordNet sentiment scorer in order to include only Verbs, Adverbs, Adjectives and Nouns that had a sentiment score greater than 0.7. Then, repeated the same representation without using the WordNet sentiment scorer to filter out the verbs, adverbs, adjectives, and nouns that had a positive or negative score that is greater than 0.7. The next step was to sample the test sets for each target domain in a similar way to that of the test sets sampled in Chen et al. (2018) in order to provide the basis for a fair comparison against our ATLAS approach. There were two types of test sets sampled for each target domain. There were a total of 20 domains in the Chen et al. (2018) dataset and each of the 20 domains was considered a target domain when experimenting with Chen et al. (2018) approach and our ATLAS approach. The two types of test sets sampled for each domain were a 200 review balanced test set that consists of 100 positives and 100 negative reviews. While the other type is an unbalanced test set where the distribution of negative to positive samples is equal to the distributions laid out in the following figure provided by Chen et al. (2018) for each domain. We have sampled the unbalanced test sets in each domain such that the negative to positive samples distribution is equal to the distributions shown in the following figure. In the next section, we will cover our transfer learning ATLAS algorithm's results averaged across the 20 domains with and without using the WordNet sentiment scorer. Subsequently, we will show our results when we convert our ATLAS algorithm from a transfer learning approach to a Lifelong learning approach.

Alarm Clock	30.51	Flashlight	11.69	Home Theater System	28.84	Projector	20.24
Baby	16.45	GPS	19.50	Jewelry	12.21	Rice Cooker	18.64
Bag	11.97	Gloves	13.76	Keyboard	22.66	Sandal	12.11
Cable Modem	12.53	Graphics Card	14.58	Magazine Subscriptions	26.88	Vacuum	22.07
Dumbbell	16.04	Headphone	20.99	Movies TV	10.86	Video Games	20.93

Figure 4-e: Fraction of negative reviews in each of the 20 domains. A figure appeared in Chen et al. (2018)

We have created a dictionary of term counts for each domain. The terms included in the dictionary had the Verb, Adverb, Adjective and Noun Part of Speech tags (POS tags). After the word count dictionary was created for each domain. The distance from the source to the target was calculated by calculating the distance from a source domain dictionary to the dictionary of the target domain's test set (sampled from the 1000 samples) in addition to the remaining target domain samples that were not included in the test set which means that the full 1000 review target data set was used in the distance calculation. At test time, the full test set is combined with the remaining samples that were not included in the test set to create the term count dictionary then the distance from each of the 19 source domains to the target domain is computed. These distances are then normalized using the Euclidean Norm function mentioned covered earlier in the methodologies chapter and each of these 19 normalized distances was assigned as weights to the 19 source domain classifiers where they were combined using the ATLAS algorithm. We will showcase the results with the balanced test set then with the unbalanced test set.

Chen et al. (2018) used the accuracy as an evaluation metric on the balanced test set and the F1score as the evaluation metric on the unbalanced test set. Therefore, we will show the average accuracy of our system on the balanced test sets across the 20 target domains. In addition to the average F1Score of our system on the unbalanced test, sets averaged across all 20 domains. The average accuracy for each target domain is calculated by randomly generating five balanced and five unbalanced test sets for that particular domain. Afterward, the ATLAS system was applied on the five balanced test sets, the accuracy on

each of these five test sets is recorded then averaged to get the average accuracy on that target domain, the same process is repeated on the remaining 19 domains to store a total of 20 average accuracies. The mean of these 20 average accuracies is computed to get the mean average-accuracy across all domains when testing on the balanced test sets. Similarly, the unbalanced test sets were generated with the exact negative to positive class distribution as in Chen et al. (2018). For each domain five unbalanced test sets were sampled according to Chen et al. (2018) then the ATLAS is applied and the F1score on each of the five test sets were recorded. Afterward, the average of the 5 F1_Scores was calculated and recorded for each domain. The process is repeated on the remaining 19 domains to get 20 average F1_Scores. The mean of these 20 average F1_Scores was computed and that is the mean average-F1_Score of the ATLAS system on the 20 different domains included in Chen et al. (2018). The F1_Scores were calculated in the same way as in Chen et al. (2018) by considering the negative label as the positive class because the negative label is the minority class in the unbalanced test sets. We have experimented with using a term frequency representation that counted the frequency of terms that had Verb, Adverb, Adjective, and Noun POS tags in addition to a second representation that computed the frequency of Verb, Adverb, Adjective and Noun terms that had a WordNet positive or negative sentiment score of greater than 0.8. We have experimented with both representations and the results are shown below.

The following four tables show the best mean of the average-accuracies recorded across the 20 domains in the Chen et al. (2018) datasets. Tables 4-yy and 4-zz show the mean of the average-accuracies and the mean of the average-f1scores recorded across the 20 domains when using the Euclidean distance based ATLAS on the unbalanced test sets.

Table 4-yy shows the results when using the WordNet sentiment scorer, and Table 4-zz shows the results when not using the WordNet sentiment scorer. We observed that the best mean of the Average-F1Scores recorded in both tables was equal to 0.7203 at a threshold of 0.6. The best average F1Score recorded in the Chen et al. (2018) paper when considering the negative label as the positive label was 0.67. Tables 4-aaa and 4-bbb show the best mean of the average-accuracies and the best mean of the average-F1scores recorded when using the Cosine distance metric. The results were identical to the results presented in tables 4-yy and 4-zz. The reason for that is in order to maintain the negative to positive samples distribution, due to the limited number of positive and negative samples, we had to use all the positive samples in each domain and randomly under sample the negative labeled samples in each domain. Since the number of negative samples in the 20 target domains were in the range of 100 to 300, the under-sampling resulted in creating five unbalanced test sets that were not so different from each other. Another way of generating random test sets was to randomly under sample the positive labeled samples but that meant that we had to undersample the negative samples even more which would have resulted in having test sets with the same negative to positive distribution but with a smaller number of samples. We wanted to sample the test sets in a way such that the negative to positive label distributions is the same as in Chen et al. (2018) with keeping the test sample as large as possible. Therefore, the only way for us to randomly sample more test sets while keeping the same negative to positive label distributions and keeping the test sets as large as possible was to keep the positive samples in each domain and undersample the negative labeled samples. Furthermore, the Euclidean and Cosine distance helped in producing the same mean average f1-score of 0.7203 across the 20 different domains as the length of the

source and target domains were equal. Therefore, the Euclidean distance metric did not get affected by the different source and target domain sizes which could have affected the final weights assigned to the source classifiers. In contrast, the results on the balanced datasets in tables 4-ccc to 4-fff had small variations in the mean average-accuracies and in the mean average-f1scores computed across the domains when comparing the results recorded when using the cosine distance vs, the results achieved when using the Euclidean distance. Tables 4-yy to 4-bbb show that the maximum mean-average f1score recorded on the unbalanced test sets was 0.7203 while treating the negative label as the positive class. The mean-average f1score was computed by applying the ATLAS algorithm on each of the 20 domains in the Chen et al. (2018) dataset by considering each of these 20 domains as the target domain. For each target domain, the ATLAS algorithm combines the 19 classifiers trained on their respective domains by calculating their weights to the target domain's unlabeled test set in addition to the target domain's remaining unlabeled samples as we have mentioned in detail earlier. The mean average-f1score is calculated by computing the average f1score achieved by the ATLAS algorithm on the five randomly generated test sets of each domain. Then recording the average f1score for each domain to end up storing 20 average f1scores. Later on, the average of all 20 average f1scores was computed to get the mean-average f1score. Chen et al. (2018) computed the average f1score across all domains by calculating the f1score for one test set per domain instead of the average f1score of five test sets for each domain. The use of the WordNet sentiment scorer in filtering some terms did not cause a boost in the mean-average f1score recorded across the 20 domains.

Table 4-ww: F1Score with WordNet (Euclidean)

Unbalanced Euclidean TL - ATLAS - with Wordnet		
Threshold	Mean-average accuracy	Mean-average f1score
0.6	0.9011	0.7203

Table 4-xx: F1Score without WordNet (Euclidean)

Unbalanced Euclidean TL-ATLAS without Wordnet		
Threshold	Mean-average accuracy	Mean-average f1score
0.6	0.9011	0.7203

Table 4-yy: F1Score with WordNet (cosine)

Unbalanced Cosine TL - ATLAS with Wordnet		
Threshold	Mean-average accuracy	Mean-average f1score
0.6	0.9011	0.7203

Table 4-zz: F1Score without WordNet (Cosine)

Unbalanced Cosine TL - ATLAS without Wordnet		
Threshold	Mean-average accuracy	Mean-average f1score
0.6	0.9011	0.7203

The following tables show the ATLAS results on the balanced test sets. The mean-average accuracy and the mean-average f1score were recorded in a similar manner to the results recorded for the unbalanced test sets. The ATLAS was tested on 20 different target domains. For each target domain, the ATLAS was tested on five randomly sampled test sets. The average accuracy and the average f1score were computed on these five randomly generated test sets. The process was repeated on the remaining 19 domains until we ended up with 20 average accuracies and 20 average f1scores across the 20 domains. The simple mean of the average f1scores was computed to get the mean-average f1score. Likewise, the simple mean of the 20 average accuracies across the 20 domains was computed to get the mean-average accuracy across the 20 domains of the Chen et al. (2018). The results are shown in tables 4-ccc through 4-fff below. The best mean-average accuracy recorded when using the WordNet sentiment scorer was 0.8641 when setting the threshold to 0.7 and using the cosine distance/similarity. The best mean-average accuracy recorded when using the WordNet sentiment scorer was 0.8635 when using the WordNet sentiment scorer and setting the threshold to 0.7. The WordNet sentiment scorer did not provide a significant boost in performance. In the following section, we will discuss the tweak that could transform our transfer learning algorithm into a lifelong learning algorithm.

Table 4-aaa: Mean average accuracies and F1Scores across the 20 domains (Euclidean distance) - With WordNet

Balanced Euclidean TL - ATLAS with Wordnet		
Threshold	Mean-average accuracy	Mean-average f1score
0.5	0.7882	0.8224
0.6	0.8383	0.8527
0.7	0.8615	0.859
0.8	0.8164	0.7817
0.9	0.6707	0.508

Table 4-bbb: Mean average accuracies and F1Scores across the 20 domains (Euclidean distance) - Without WordNet

Balanced Euclidean TL - ATLAS without Wordnet		
Threshold	Mean-average accuracy	Mean-average f1score
0.5	0.7866	0.8209
0.6	0.8377	0.852
0.7	0.8584	0.8553
0.8	0.815	0.7794
0.9	0.6726	0.5105

Table 4-ccc: Mean average accuracies and F1Scores across the 20 domains (cosine distance) - With WordNet

Balanced Cosine TL - ATLAS with Wordnet		
Threshold	Mean-average accuracy	Mean-average f1score
0.5	0.7899	0.8232
0.6	0.8404	0.8544
0.7	0.8641	0.8619
0.8	0.817	0.7822
0.9	0.6696	0.5053

Table 4-ddd: Mean average accuracies and F1Scores across the 20 domains (cosine distance) - Without WordNet

Balanced Cosine TL - ATLAS without Wordnet		
Threshold	Mean-average accuracy	Mean-average f1score
0.5	0.7902	0.8236
0.6	0.8424	0.8562
0.7	0.8635	0.86047
0.8	0.8183	0.7839
0.9	0.6721	0.5108

LIFELONG LEARNING ATLAS

The ATLAS algorithm could be tweaked to become a lifelong learning sentiment classification method. Instead of measuring the distance from each source domain's samples to the target domain's combined full test set and the remaining unlabeled samples of the target domain, the distance measurement is performed at test time, one test review at a time instead. Before calculating the weights associated with each classifier, before testing. The weights are calculated when receiving the test review as input to the system. The distance from that target test review to all 19 source domains' samples is measured in the manner described earlier in the methodologies and results chapter. Then the 19 distances are normalized and these normalized distances act as the weights to the classifiers trained on their corresponding source domains. The ATLAS algorithm is then applied using these calculated weights when classifying the input test review. Afterward, the process is repeated for each test review in the test set. The time complexity added by this tweak is huge. Therefore, we have experimented with the classification threshold that resulted in the best F1Score and Accuracy across all domains which is equal to 0.6. Tables 4-ggg and 4-hhh show the average F1scores recorded for the unbalanced test sets when using the WordNet and when not using the WordNet sentiment scorer. The results that were recorded showed that using the WordNet sentiment scorer when filtering the Verbs, Adverbs, Adjectives, and Nouns that had a positive or negative sentiment score of less than 0.7 did not boost performance. As it did not help in boosting the overall average F1score or average accuracy across the target domains in the Transfer Learning or in the Lifelong learning modes. The best F1-Score while considering the negative label as the positive class recorded in Chen et al. (2018) was 67%. In our results, the best average F1-Score for the

negative label as the positive class recorded across all domains when using a threshold of 0.6 was 72.56% when using the Euclidean distance measure and 72.07% when using the Cosine distance measure at the Lifelong learning mode of the ATLAS. As for the balanced datasets, tables 4-iii and 4-jjj show the average test set accuracy on the balanced test set which is still in the range of the average balanced accuracy recorded for the transfer learning version of the ATLAS algorithm shown earlier. Even though WordNet did not help with boosting the overall average f1score or average accuracy, they helped in decreasing the processing time of our algorithm without hurting the average F1score or average accuracy achieved.

Table 4-eee: Best average F1Score (Euclidean LL)

Unbalanced Euclidean LL - ATLAS - with Wordnet		
Threhold	average accuracy	average f1score
0.6	0.902	0.7255

Table 4-fff: Best average F1Score (Cosine LL)

Unbalanced Cosine LL - ATLAS with Wordnet		
Threhold	average accuracy	average f1score
0.6	0.9	0.7207

Table 4-ggg: Best average Accuracies (Euclidean LL)

Balanced Euclidean LL - ATLAS - with Wordnet		
Threhold	average accuracy	average f1score
0.6	0.8455	0.8606
0.7	0.8665	0.8666

Table 4-hhh: Best average Accuracies (Cosine LL)

Balanced Cosine LL - ATLAS with Wordnet		
Threhold	average accuracy	average f1score
0.6	0.8425	0.858
0.7	0.8645	0.8648

Figures 12 shows the average accuracy reported in Chen et al. (2018) on the balanced test sets across the 20 domains that was equal to 0.8334 (83.34%) and the mean of all the average accuracies recorded by the lifelong learning mode of the ATLAS algorithm across the 20 domains at a threshold of 0.7 which was equal to 0.8665 (86.65%) as shown in Table 4-iii. Figure 11 shows the average F1Score recorded for the Chen et al. (2018) lifelong learning algorithm across the unbalanced test sets of the 20 domains versus the mean of the average F1Score recorded for the ATLAS algorithm while on the lifelong learning mode. Figures 9 and 10 show the average accuracies across the 20 domains on the balanced

test sets in addition to the average F1Scores across the 20 domains on the unbalanced test sets when applying the ATLAS algorithm in a Transfer learning mode. We have showcased our results against the results reported in Chen et al. (2018). The average F1Score across the 20 domains reported by Chen et al. (2018) was 0.6700 (67.00%) while the average of all the average F1Scores recorded for the 20 target domains using the lifelong learning mode of the ATLAS algorithm at a threshold of 0.6 was 0.7255 (72.55%) which is a significant improvement on what has been reported in Chen et al. (2018). Figure 9 shows the average F1Score reported when using the transfer learning mode of the ATLAS algorithm when using WordNet and when not using the WordNet sentiment scorer. These results show that WordNet could be helpful but not significantly helpful.

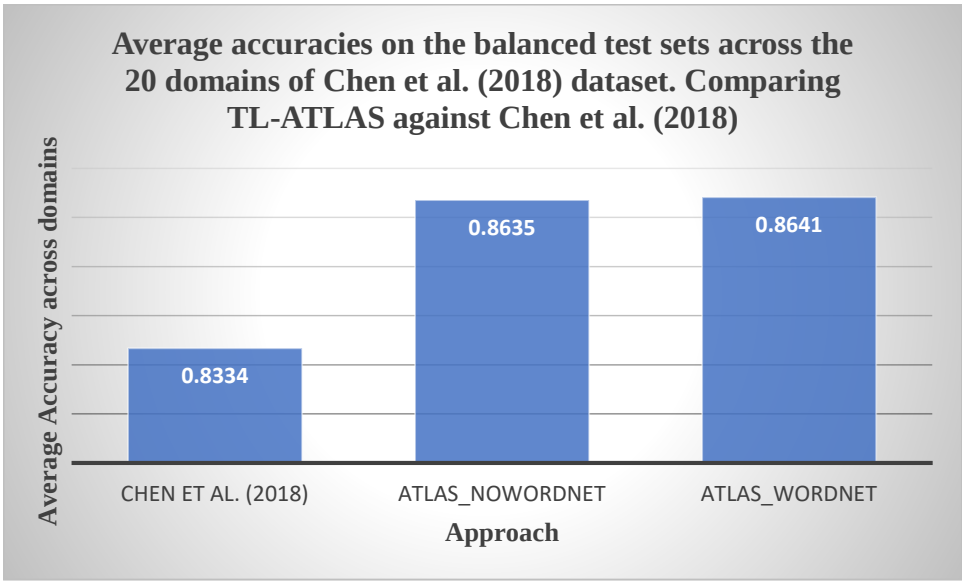

Figure 4-g: Average accuracies across the 20 domains of Chen et al. (2018) Transfer Learning-ATLAS vs Chen et al. (2018)

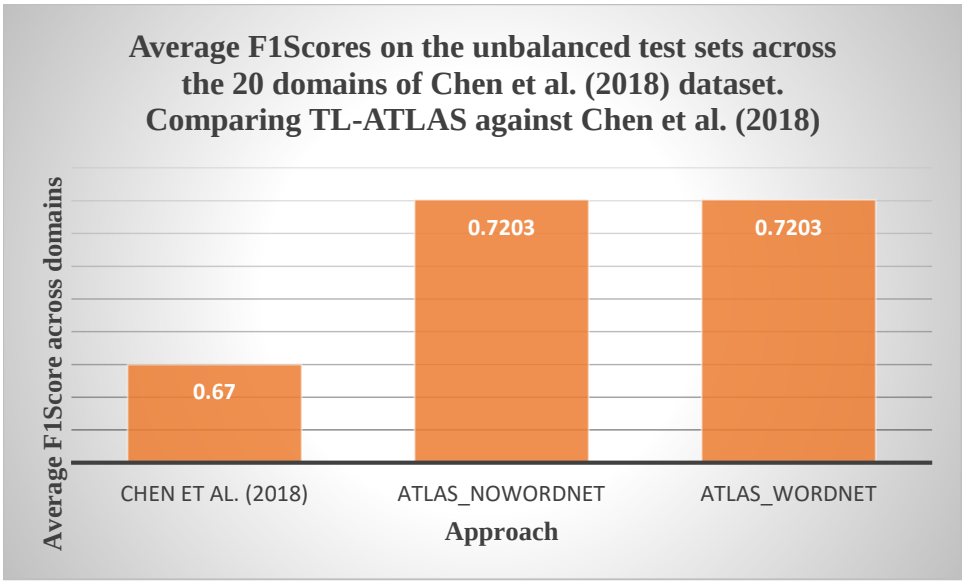

Figure 4-f: Average F1Scores across the 20 domains of Chen et al. (2018) Transfer Learning-ATLAS vs Chen et al. (2018)

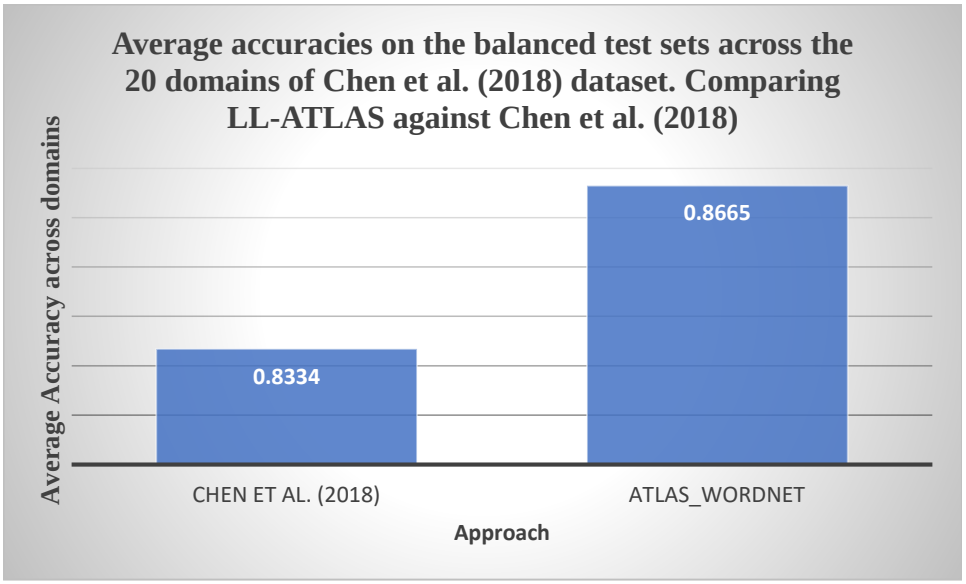

Figure 4-i: Average accuracies across the 20 domains of Chen et al. (2018) Lifelong Learning-ATLAS vs Chen et al. (2018)

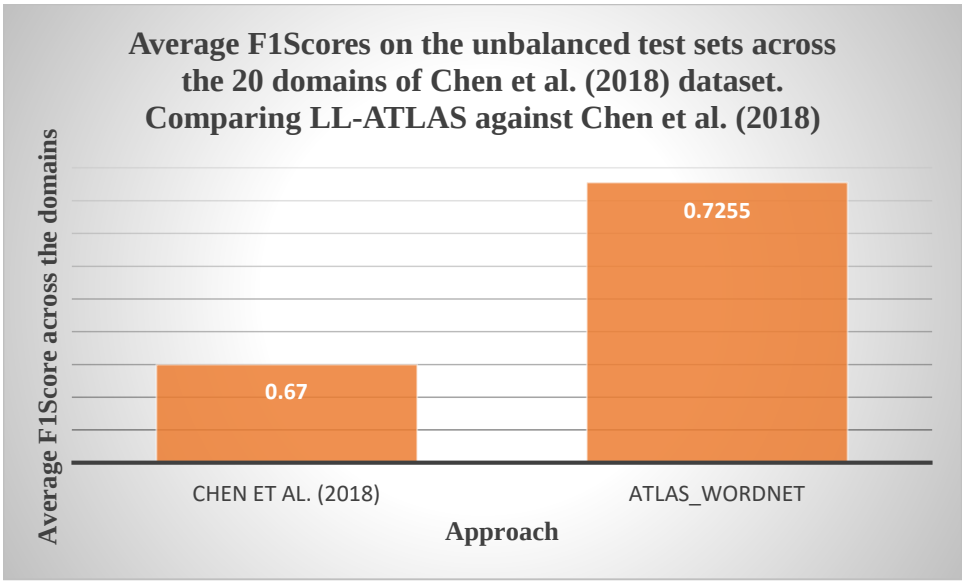

Figure 4-h: Average F1Scores across the 20 domains of Chen et al. (2018) Lifelong Learning-ATLAS vs Chen et al. (2018)

DEEP TEXT GENERATION RESULTS

In this section, we will showcase the results when using deep learning text generators to generate our training sets for four of our test domains of the Blitzer et al. (2007) datasets Books, Kitchen, DVD, and Electronics. These results reflect how creating training sets could enhance the training process. We have started comparing between deep learning techniques vs Markov chain text generation techniques when generating training samples in the target domain and we have published our results in Abdelwahab et al. (2018) paper titled "deep learning based vs Markov chain based text generation for sentiment classification". In this paper, we observed that deep learning text generators generated samples that boosted our end classifier's accuracy and F1Score on the target test sets when compared against Markov chain text generators. The deep learning text generation techniques even proved to be better at generating useful samples when even trained on noisy/weakly labeled samples when compared against Markov Chain text generators. Weakly labeled samples which were labeled using a rule-based sentiment classifier that had an accuracy of 71% on a balanced test set. In the case where no labeled samples are provided, and the rule-based classifier is used in labeling the seed samples (initial samples that were used in training the deep learning text generators and the Markov chain text generators), the deep learning text generators proved to be better at producing samples from these weakly labeled seed samples when compared against the Markov chain text generators.

The evaluation criteria for the text generation methods will be task oriented. Which means that the quality of the generated text will be evaluated based on the best accuracy achieved

by the end classifier after grid search on the balanced kitchen test set. We will show the best accuracy achieved with each generated training set, L1 and L2 penalty combinations on the balanced Kitchen test set of the Blitzer et al. (2007). All F1Scores and accuracy values were averaged over 10 trials.

Table 4-iiiml: Accuracy and F1Score on the Kitchen test set provided by Bollegala et al. (2015)

Training Data	L1	L2	Acc	F-Score
GRU_Full	10	100000	0.74	0.75
LSTM_Full	100	100000	0.77	0.74

Table 4-jjj: Accuracy and F1Score on the Kitchen test set provided by Bollegala et al. (2015) when using Markov Chain generators

Training Data	L1	L2	Accuracy
Markov_order1	100	100	0.54
Markov_order10	10	1000	0.502

Table 4-kkk: Accuracy and F1Score on the Kitchen test set provided by Bollegala et al. (2015)when using Markov Chain text generators trained on 100% correctly labeled seed reviews

Training Data	L1	L2	Accuracy
Markov_order1	1000	100000	0.71
Markov_order10	10	10	0.72

Tables 4-kkkml and 4-lll show that the deep learning based text generators resulted in better performance when trained on the noisy/unsupervised labeled seed samples when compared against the Markov chain text generators. However, Table 4-mmm shows that when training the Markov chain text generators on 100% correctly labeled samples, it performs way better than when it is trained on weakly labeled samples which means that deep learning based text generators proved to be more resilient to poorly labeled training samples. After observing these results, we have concluded to try different techniques which lead us to develop our algorithm, the Adaboost Inspired Transfer Learning Approach for Sentiment Classification (ATLAS).

CONCLUDING THE DEEP TEXT GENERATION RESULTS

The results that we have obtained suggest that Markov Chain based text generators have little tolerance to incorrectly labeled reviews in the training set. As the accuracy of our unsupervised rule-based classifier is around 71% on a balanced test set. Therefore, a percentage of the labeled seed reviews contain false positives or false negatives which lead to damping the performance of the Markov chain text generators as we have shown in the results section. On the other hand, deep learning based models have higher resilience towards the presence of false positives or false negatives in the training set which showed that the accuracies and F1Scores achieved by the end classifier did not dip to the fifties as in the case of using the Markov chain text generators. We will explore using text generation on a larger training set in the future while exploring other Neural networks architectures like the GAN networks. We will explore using text generators that make use of word level and character level features as Xie et al. (2017) have discovered that language models that make use of character level features and word level features perform better than models that depend on character level feature only or word level features only.

CHAPTER V

CONCLUSION

A classifier trained on a source domain will perform poorly when applied on a totally different domain. On the other hand, a classifier trained on a source domain will perform acceptably on a target domain that is slightly different from it. Which led us to think that if we have classifiers trained on k different domains then there should be a way to aggregate their predictions on the target domain in a way to achieve a high accuracy and F1Score when compared against other transfer learning or lifelong learning algorithms. We faced a hurdle when inferring the weights to these classifiers as in Adaboost, the weights were computed from the classifiers' training error while training. Since we do not use any labeled samples in the target domain, we had to figure out a way to assign weights to the source classifiers where we get to aggregate their predictions on the target domain's test sets efficiently. So, we have decided to observe whether domain similarities could be used in computing weights for the source domain classifiers and we could say after the observations recorded above that using the domain similarities between the source domains to the target domains as weights to the source domain classifiers has boosted the accuracies and F1Scores of our system on two benchmark datasets when compared to other techniques published in 2017 and 2018. The results show that domain similarity is helpful can be used in computing weights for the source domain classifiers. The domain similarity metrics we have used in ATLAS were not sophisticated and there are other sophisticated techniques

in computing domain similarities as in Wu et al. (2016) but we wanted to test with simple techniques to show the potential of using more advanced techniques in our future work. The major limitation of the ATLAS algorithm is when having classifiers trained on domains that are away from the target domain. An example of this case was shown earlier when we recorded the accuracies and F1Scores of the least performing classifier groups. These classifiers' accuracies and F1Scores were not boosted to the point that makes ATLAS feasible. Thus, ATLAS makes use of classifiers that are trained on domains that are not far away from the target domain. As for text generation, we will experiment with pre-trained text generators instead of text generators trained on a small sample of in-domain reviews.

REFERENCES

Tao Li, Vikas Sindhwani, Chris Ding, and Yi Zhang. Bridging Domains with Words: Opinion Analysis with Matrix Tri-factorizations. In Proceedings of SIAM, 2013.

Richard Socher, Milind Ganjoo, Christopher D. Manning, and Andrew Y. Ng. Zero-Shot Learning Through Cross-Modal Transfer. In Proceedings of NIPS, 2013.

Kun-Lin Liu, Wu-Jun Li, and Minyi Guo. Emoticon Smoothed Language Models for Twitter Sentiment Analysis. In Proceedings of AAAI, 2012.

Songbo Tan and Xueqi Cheng. Improving SCL Model for Sentiment-Transfer Learning. In Proceedings of NAACL HLT 2009, Boulder, Colorado, June 2009.

Pedro H. Calais Guerra, Adriano Veloso, Wagner Meira Jr., and Virgilio Almeida. From Bias to Opinion: a Transfer-Learning Approach to Real-Time Sentiment Analysis. In Proceedings of KDD'11, August 21-24, 2011, San Diego, California, USA.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA, USA, 2011.

Songbo Tan, Xueqi Cheng, Yuefen Wang, and Hongbo Xu. Adapting Naïve Bayes to Domain Adaptation for Sentiment Analysis. In Proceedings of ECIR, 2009.

Lei Zhang, Riddhiman Ghosh, Mohamed Dekhil, Meichun Hsu, and Bing Liu. Combining Lexicon-based and Learning-based Methods for Twitter Sentiment Analysis. In proceedings of HP Laboratories, HPL-2011-89, 2011.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye The. A fast learning algorithm for deep belief nets. In Proceedings of Neural Computation 2006.

Tao Li, Vikas Sindhwani, Chris Ding, and Yi Zhang. Knowledge Transformation for Cross-Domain Sentiment Classification. In Proceedings of SIGIR'09, July 19-23, 2009, Boston, Massachusetts, USA.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Cross-Domain Sentiment Classification via Spectral Feature Alignment. In Proceedings of WWW 2010, April 26-30, 2010, Raleigh, North Carolina, USA.

Testsuya Nasukawa and Jeonghee Yi. Sentiment Analysis: Capturing Favorability Using Natural Language Processing. In proceedings of K-CAP'03, October 23-25, 2003, Sanibel Island, Florida, USA.

Kun-Lin Liu, Wu-Jun Li, and Minyi Guo. Emoticon Smoothed Language Models for Twitter Sentiment Analysis. In Proceedings of the AAAI conference, 2012.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. In Proceedings of EMNLP 2002.

Yoshua Bengio. Deep Learning of Representations for Unsupervised and Transfer Learning. In proceedings of JMLR, 2012.

Hal Daume III, Abhishek Kumar, and Avishek Saha. Co-regularization Based Semi-Supervised Domain Adaptation. In proceedings of the DANLP workshop at ACL 2010.

John Blitzer, Mark Dredze and Fernando Pereira. Biographies, Bollywood, Boom-boxes, and Blenders: Domain Adaptation for Sentiment Classification. In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, Prague, Czech Republic, June 2007.

Kerstin Denecke. Are SentiWordNet Scores Suited for Multi-Domain Sentiment Classification? IEEE, 2009.

Boqing Gong, Kristen Grauman, and Fei Sha. Connecting the Dots with Landmarks: Discriminatively Learning Domain-Invariant Features for Unsupervised Domain Adaptation. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013.

Mike Thelwall and Kevan Buckley. Topic-Based Sentiment Analysis for the Social Web: The role of Mood and Issue-Related Words. In Proceedings of the Journal of the American Society for Information Science and Technology, 2012.

Shoushan Li, Yunxia Xue, Zhongqing Wang, and Guodong Zhou. Active Learning for Cross-Domain Sentiment Classification. In Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, 2013.

Jihie Kim, Jaebong Yoo, Ho Lim, Huida Qiu, Zornitsa Kozareva, and Aram Galstyan. Sentiment Prediction Using Collaborative Filtering. In the proceedings of the Seventh International AAAI Conference on Weblogs and Social Media, 2013.

Christian Scheible and Hinrich Schutze. Sentiment Relevance. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Sofia, Bulgaria, August 4-9 2013.

Danushka Bollegala, David Weir and John Carroll. Using Multiple Sources to Construct a Sentiment Sensitive Thesaurus for Cross-Domain Sentiment Classification. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, Portland, Oregon, June 19-24, 2011.

Natalia Ponomareva and Mike Thelwall. Do Neighbors Help? An Exploration of Graph-based Algorithms for Cross-domain Sentiment Classification. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Jeju Island, Korea, 12-14 July 2012.

Bin Wei and Christopher Pal. Cross Lingual Adaptation: An Experiment on Sentiment Classifications. In Proceedings of the ACL 2010 Conference Short Papers, Uppsala, Sweden, 11-16 July 2010.

Rui Xia and Chengqing Zong. A POS-based Ensemble Model for Cross-domain Sentiment Classification. In Proceedings of the 5th International Joint Conference on Natural Language Processing, Chiang Mai, Thailand, November 8-13, 2011.

Kang Liu and Jun Zhao. Cross-Domain Sentiment Classification Using a Two-Stage Method. In Proceedings of CIKM'09, November 2-6, 2009, Hong Kong China.

Shusen Zhou, Qingcai Chen and Xiaolong Wang. Active Deep Networks for Semi-Supervised Sentiment Classification. In Proceedings of Coling 2010, Beijing, August 2010.

Zhiyuan Chen, Nianzu Ma, and Bing Liu. Lifelong Learning for Sentiment Classification. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Conference on Natural Language Processing (Short Papers), Beijing, China, July 26-31, 2015.

Hui Yang, Luo Si, and Jamie Callan. Knowledge Transfer and Opinion Detection in the TREC2006 Blog Track. In Proceedings of the Journal of Information Retrieval. 2006.

Songbo Tan, Gaowei Wu, Huifeng Tang, and Xueqi Cheng. A novel scheme for domain-transfer problem in the context of sentiment analysis. In Proceedings of the 16th ACM conference on Conference on information and knowledge management, Lisbon, Portugal, ACM. 2007.

John Blitzer, Ryan McDonald, and Fernando Pereira. Domain Adaptation with Structural Correspondence Learning. In Proceedings of the EMNLP Conference, 2006.

Hal Daume III. Frustratingly Easy Domain Adaptation. In proceedings of ACL 2007.

Yulan He, Chenghua Lin, and Harith Alani. Automatically Extracting Polarity-Bearning Topics for Cross-Domain Sentiment Classification. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, Portland, Oregon, June 19-24, 2011.

Alina Andreevskaia and Sabine Bergler. When Specialists and Generalists Work Together: Overcoming Domain Dependence in Sentiment Tagging. In proceedings of ACL-08, Columbus, Ohio, USA, June 2008.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. Unsupervised Cross-Domain Word Representation Learning. In proceedings of ACL 2015.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

Andrew B. Goldberg and Xiaojin Zhu. Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization. In Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing (TextGraphs '06). 2006.

Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval. Vol. 2, 2008.

Fabrizio Sebastiani and Andrea Esuli. SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. 2006.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-Based Methods for Sentiment Analysis. In Proceedings of Association for Computational Linguistics, 2010.

Gerlof Bouma. Normalized (pointwise) mutual information in collocation extraction. In Proceedings of the Biennial GSCL Conference 2009.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S., and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In Burges, C.j.c., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K.q. (eds.), Advances in Neural Information Processing Systems, 2013.

Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In Proceedings of the 31st International Conference on Machine Learning, Beijing, China. 2014.

Carlos Guestrin and Emily Fox. Machine Learning Foundations: A Case Study Approach. By the University of Washington on Coursera, Seattle, WA. 2015.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M Mohammed, Alan Ritter, and Veselin Stoyanov. SemEval 2015 Task 10: Sentiment Analysis in Twitter. In Proceedings of the 9th International Workshop on Semantic Evaluation (SEMEVAL 2015). Association for Computational Linguistics, Denver, Colorado. 2015.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Fabrizio Sebastiani. 2016. SemEval-2016 Task 4: Sentiment Analysis in Twitter. In Proceedings of the 10th International Workshop on Semantic Evaluation (SEMEVAL 2016). Association for Computational Linguistics, San Diego, California. 2016.

Bing Liu and Minqing Hu. Mining and Summarizing Customer Reviews. In Proceedings of KDD'04, August 22-25, 2004, Seattle, Washington, USA. 2004.

Zhe Zhang and Munindar P. Singh. ReNew A Semi-Supervised Framework for Generating Domain-Specific Lexicons and Sentiment Analysis. In Proceedings of the

52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, Maryland, USA, June 23-25 2014.

Xiang Zhang and Yann LeCun. Text Understanding from Scratch. In arXiv: 1502.01710v5 [cs.LG] 4 Apr 2016.

Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. Mining of Massive Datasets. 2014.

Cane Wing-Ki Leung, Stephen Chi-Fai Chan, and Fu-lai Chung. Integrating Collaborative Filtering and Sentiment Analysis: A Rating Inference Approach. In proceedings of The ECAI 2006 Workshop on Recommender Systems. 2006.

Ronan Collobert, Koray Kavukcuoglu, and Clement Farabet. Torch7: A Matlab-like environment for machine learning. In Proceedings of BigLearn, NIPS Workshop, number EPFL-CONF-192376, 2011a.

Zhiyuan Chen, Nianzu Ma, and Bing Liu. Lifelong Learning for Sentiment Classification. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015). 2015.

Zhiyuan Chen, Nianzu Ma, and Bing Liu. Lifelong Learning for Sentiment Classification. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015). arXiv:1801.02808 2018.

Fangzhao Wu and Yongfeng Huang. Sentiment Domain Adaptation with Multiple Sources. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 301–310, Berlin, Germany, August 7-12, 2016.

Fangzhao Wu, Yongfeng Huang, and Jun Yan. Active Sentiment Domain Adaptation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1701–1711 Vancouver, Canada, July 30 - August 4, 2017

Omar Abdelwahab and Adel Elmaghraby. Effect of training set size on SVM and Naive Bayes for Twitter sentiment analysis. In proceedings of 2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT).

Omar Abdelwahab and Adel Elmaghraby. Deep Learning Based vs. Markov Chain Based Text Generation for Cross-Domain Adaptation for Sentiment Classification. 2018 IEEE International Conference on Information Reuse and Integration (IRI)

George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

Fangzhao Wu, Yongfeng Huang, and Jun Yan, Active Sentiment Domain Adaptation, Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1701–1711 Vancouver, Canada, July 30 - August 4, 2017.

Jianfei Yu, and Jing Jiang, Learning Sentence Embeddings with Auxiliary Tasks for Cross-Domain Sentiment Classification, Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, Texas, November 1-5, 2016.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. Proceedings of WWW '10 Proceedings of the 19th international conference on World wide web Pages 751-760.

Natalia Ponomareva and Mike Thelwall. 2012. Do neighbours help?: an exploration of graph-based algorithms for cross-domain sentiment classification. EMNLP-CoNLL '12 Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning Pages 655-665.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015 Unsupervised Cross-Domain Word Representation Learning. Proceedings of e 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 730–740.

Omar Abdelwahab and Adel Elmaghraby. 2016. UofL at SemEval-2016 Task 4: Multi Domain word2vec for Twitter Sentiment Classification. Proceedings of SemEval-2016, pages 169–175

Xiang Zhang, and Yann LeCun. 2016. Text Understanding from Scratch. arXiv:1502.01710

Sean Robertson GitHub. (2017). spro/char-rnn.pytorch. [online] Available at: https://github.com/spro/char-rnn.pytorch

Stanley Xie, Ruchir Rastogi, and Max Chang. 2017. Deep Poetry: Word-Level and Character-Level Language Models for Shakespearean Sonnet Generation. [online] Available at: https://web.stanford.edu/class/cs224n/reports/2762063.pdf

Gamon M., Aue A., Corston-Oliver S., Ringger E. (2005) Pulse: Mining Customer Opinions from Free Text. In: Famili A.F., Kok J.N., Peña J.M., Siebes A., Feelders A. (eds) Advances in Intelligent Data Analysis VI. IDA 2005. Lecture Notes in Computer Science, vol 3646. Springer, Berlin, Heidelberg

Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M. Hellerstein. 2012. Distributed GraphLab: a framework for machine learning

and data mining in the cloud. Proc. VLDB Endow. 5, 8 (April 2012), 716-727. DOI: https://doi.org/10.14778/2212351.2212354

CURRICULUM VITA

Name:	Omar Abdelwahab
Address:	Duthie Center of Engineering 222 Eastern Parkway University of Louisville Louisville, KY 40208
DOB:	Cairo, Egypt - May 22nd, 1988
Education and Training:	B.S., Computer Engineering Cairo University 2005-2010 MS, Computer Science University of Louisville 2012-2013 Ph.D., Computer Science and Engineering University of Louisville
AWARDS:	IEEE best student in CECS award – 2016
INTERNSHIPS:	NLP/Machine Learning Intern, 3M – 2017 Research Intern , IBM – 2014
Publications:	Effect of training set size on SVM and Naive Bayes for Twitter sentiment analysis UofL at SemEval - 2016 Task 4: Multi Domain word 2vec for Twitter Sentiment Classification Evolution of a Metaheuristic for Aggregating Wisdom from Artificial Crowds Deep Learning Based vs. Markov Chain Based Text Generation for Cross Domain Adaptation for Sentiment Classification