Cross-domain Sentiment Classification using an Adapted Naïve Bayes Approach and Features Derived from Syntax Trees

Srilaxmi Cheeti, Ana Stanescu and Doina Caragea
Department of Computing and Information Sciences, Kansas State University, Manhattan, KS, U.S.A.

Keywords: Domain Adaptation, Sentiment Classification, Adapted Naïve Bayes, Syntax Trees.

Abstract: Online product reviews contain information that can assist in the decision making process of new customers looking for various products. To assist customers, supervised learning algorithms can be used to categorize the reviews as either positive or negative, if large amounts of labeled data are available. However, some domains have few or no labeled instances (i.e., reviews), yet a large number of unlabeled instances. Therefore, domain adaptation algorithms that can leverage the knowledge from a source domain to label reviews from a target domain are needed. We address the problem of classifying product reviews using domain adaptation algorithms, in particular, an Adapted Naïve Bayes classifier, and features derived from syntax trees. Our experiments on several cross-domain product review datasets show that this approach produces accurate domain adaptation classifiers for the sentiment classification task.

1 INTRODUCTION

Web 2.0 contains a vast amount of user generated information, in the form of reviews, blogs, webpages, etc. Shoppers tend more and more to seek online reviews before making a purchase. For example, users interested in buying a camera must evaluate alternative products with various characteristics. In addition to product descriptions, positive and negative opinions from previous users can also make an impact on the customer's choices. Manufacturers and retailers also find such reviews helpful, as they can learn more about customer's likes and dislikes and adjust the products accordingly, or use that information to train recommender systems for suggesting products to potential users, and targeting customers.

Manually classifying customer reviews can be an intensive, time consuming process, as it requires a lot of browsing and reading of reviews. Therefore, automated tools to do this classification are desirable, as they could save both customers and companies a lot of time and quickly provide the gist of the reviews about a product. Automated classification of online data as positive, negative or neutral is known as sentiment classification, an area at the intersection of Machine Learning (ML) and Natural Language Processing (NLP). In supervised frameworks, the sentiment classification problem is formulated as a machine learning problem, where labeled training data is provided to a learning algorithm and a classifier is learned. The resulting classifier can then predict the sentiment of new unlabeled data. Both training and test instances are represented using automatically generated features, e.g., NLP features.

The sentiment classification, in general, can be addressed at word, sentence, or document level. Much of the previous sentiment classification work has been done at the document level using keyword based approaches, and there has not been a lot of work done at the sentence level. Sentence level classification is more challenging when compared to document level classification because classification of a sentence as positive, negative or neutral has to be performed in the absence of context. This problem can be alleviated, if two or more consecutive sentences are combined together, or if the whole document is used. Another challenge in sentiment classification is that a sentence or a document can have more than one sentiment.

In this work, we focus on sentiment classification at sentence level, but consider sentences that have only one sentiment, either positive or negative, as neutral reviews are not particularly helpful in the process of making a decision. Usually, with enough training data, the supervised approach can produce accurate domain-specific classifiers. For example, one can use movie review data to train a movie sentiment classifier and then use the classifier to predict the sentiment of new movie reviews. However, in real world ap-

Cheeti S., Stanescu A. and Caragea D..
Cross-domain Sentiment Classification using an Adapted Naïve Bayes Approach and Features Derived from Syntax Trees.
DOI: 10.5220/0004546501690176
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval and the International Conference on Knowledge Management and Information Sharing* (KDIR-2013), pages 169-176
ISBN: 978-989-8565-75-4

plications, the amount of labeled data for a particular domain can be limited and it is interesting to consider cross-domain classifiers, in other words, classifiers that leverage training data from a source domain to learn a classifier for a target domain with limited labeled data. For example, we can use books as the source domain, while the target domain can be either music, DVDs, movies, electronics, clothing, toys, etc.

Generally, a classifier built on one domain (i.e., source domain) does not perform well when used to classify the sentiment in another domain (i.e., target domain). One reason for this is that there might be some specific words that express the overall polarity of a given sentence in a given domain, and the same words can have different meaning or polarity in another domain. Let us consider kitchen appliances and cameras as our domains, then words such as *good* and *excellent* express positive sentiments in both kitchen appliance domain, as well as camera domain. Words such as *bad* and *worse* express a negative sentiment in both domains; they are known as domain independent words. On the other hand, words such as *safe, stainless, sturdy, efficient* express sentiments in the kitchen domain and may or may not express any sentiment in the camera domain. These are known as domain dependent or domain specific words.

In cross-domain classification, the general goal is to use labeled data in the source domain and, possibly, some labeled data in the target domain, together with unlabeled data from the target to learn cross-domain classifiers for predicting the sentiment of future target instances. The cross-domain sentiment classification problem presents additional challenges compared to the corresponding problem in a single domain. Using both source and target data to construct the classifier requires substantial insight and effort, specifically with respect to how to choose source features that are predictive for target, and also how to combine data or classifiers from source and target.

To address the first problem, most recent approaches (Blitzer et al., 2006), (Blitzer et al., 2007), (Tan et al., 2009) identify domain independent features (a.k.a., generalized or pivot features) to represent the source, and domain specific features to represent the target. Domain independent features serve as a bridge between source and target, thus reducing the gap between them. The performance of the final classifier will heavily depend on the domain independent features; therefore, care must be used when selecting these features. In this work, we use NLP syntax structured trees to generate features. Domain independent features are selected based on the frequently co-occurring entropy (FCE) method proposed by Tan et al. (2009). Features with high entropy values are assumed to be independent features and used to represent the source domain. Furthermore, to combine source and target data, we use an Expectation Maximization (EM) based Naïve Bayes classifier proposed also by Tan et al., (2009). Originally, the approach in (Tan et al., 2009) assumes labeled source data and unlabeled target data. In our implementation, we can also incorporate labeled target domain data, if available. As the number of iterations increases, we reduce the weight for the source domain instances, while increasing the weight for the target domain instances, so that the resulting classifier can ultimately be used for predicting target domain instances.

2 RELATED WORK

Sentiment classification across domains is a very challenging problem. Classifiers trained on one domain cannot always predict the instances from a different domain accurately, due to the fact that domain-specific features can have different meanings in different domains. The main challenges when performing sentiment classification experiments consist of selecting the appropriate features and the right Machine Learning algorithms for a particular dataset.

Relevant to our work, in the context of a single domain sentiment classification, Harb et al., (2008) introduced the AMOD (Automatic Mining of Opinion Dictionaries) approach consisting of the following three phases. The first phase, known as the Corpora Acquisition Learning Phase, solves a major challenge by automatically extracting the data from the web using a predefined set of seed words (positive and negative terms). The second phase, also known as the Adjective Extraction Phase, extracts a list of adjective words with positive and negative opinions. The third phase, known as the Classification Phase is used to classify the given documents using the adjective words extracted in the second phase. The authors used unigrams as AMOD features and then used the list of adjective words to classify the given documents. They used movie review dataset and the car dataset and the results show that the AMOD approach was able to classify the given documents by using a list of adjective words in a single domain.

Zhang et al., (2010) proposed to use several types of syntax subtrees as features, where the subtrees are obtained from complete syntax trees by using both adjective and sentiment word pruning strategies. The syntax trees are derived using the Stanford parser. These features were found to be very efficient for classification in a single domain scenario.

Blitzer et al., (2007) introduced a domain adap-

tation strategy, which is an extension of an approach previously proposed by the same authors, called structural correspondence learning (SCL) (Blitzer et al., 2006). As a baseline, the authors first chose a set of features that occur frequently in both source and target domains as pivot (or generalized) features, and compare these features with the pivot features selected as those target features that have the highest mutual information (MI) to the source domain. First, the authors assume that the source domain dataset contains labeled and unlabeled data, whereas the target domain dataset contains only unlabeled data. They observed that choosing the pivot features using MI has reduced the relative error by 36%. After introducing 50 labeled instances from the target domain, they observed that the average reduction in error is 46%. Overall, the algorithm is found to be very useful for cross-domain sentiment classification especially due to the use of the MI to select the pivot features. Furthermore, the results in this paper show that using a small number of training labeled data can yield improved classifiers.

The Spectral Feature Alignment (SFA) algorithm for cross-domain sentiment classification was proposed by Pan et al., (2010). The process of selecting the pivot features is the same as the one described in (Blitzer et al., 2007). The results of the SFA algorithm are better than SCL and NoTransf, where a classifier is trained using only source domain data.

Tan et al. (2009) proposed an Adapted Naïve Bayes (ANB) algorithm to perform cross-domain sentiment classification. The first step in their approach is to find generalized features that can serve as a bridge between the source and the target domains. In order to retrieve the generalized features they used a frequently co-occurring entropy method and picked the features with the highest entropy values as the generalized features. Subsequently, two classifiers are learned, one from the source domain using only the generalized features and the other from the target domain using all the features from the target domain. Next, the classifiers are used to predict the target domain unlabeled instances. The process of learning the classifiers and then using them to predict the target domain instances is repeated until the algorithm convergences. The authors used Chinese domain-specific datasets for their experiments. They compared the ANB algorithm with Naïve Bayes Multinomial (supervised), EM-based Naïve Bayes (semi-supervised) described in (Nigam et al., 1998), Naïve Bayes Transfer Classifier (transfer-learning) described in (Dai et al., 2007). The results show that ANB performs much better than the other algorithms.

3 ANB WITH SYNTAX TREE FEATURES

Our goal is to perform sentence level sentiment classification across domains. As described earlier, Zhang et al. (2010) have shown that features constructed based on syntax trees can give good results for sentiment classification problems in a single domain. We have also seen that there are many algorithms for learning cross-domain classifiers, including SCL (Blitzer et al., 2006), AMOD (Harb et al., 2008), SFA (Pan et al., 2010), and ANB (Tan et al., 2009). Previous results have shown that the features used and the methods for selecting generalized features, for example MI (Pan et al., 2010) or FCE (Blitzer et al., 2006), can have a high impact on the performance of the resulting classifiers. To identify generalized features, we use the frequently co-occurring entropy (FCE) proposed by Tan et al. (2009), thus eliminating the need for a predefined set of domain specific and domain independent features required by other methods. We also use the Adapted Naïve Bayes algorithm (ANB) proposed by Tan et al. (2009) to learn cross-domain classifiers. However, we have modified the ANB approach to enable it to use labeled data from the target domain during the training phase.

3.1 ANB with Target Labeled

Adapted Naïve Bayes (ANB) (Tan et al., 2009) is a domain adaptation algorithm, based on a weighted transfer version of the Naïve Bayes classifier. It builds a classifier using the Expectation Maximization (EM) technique together with the Naïve Bayes classifier, to predict the target domain unlabeled data. More specifically, the EM algorithm is used to maximize the likelihood of the data, and consists of two steps: E-step (Expectation-step) and M-step (Maximization-step). In the E-step, we estimate the missing data (in our case, the labels of the unlabeled target data) given the current model. In the M-step, we update the model by maximizing the likelihood function under the assumption that the missing data is now known. The two steps are repeated until convergence.

The EM approach used in ANB and described in (Tan et al., 2009) is different from the traditional EM, as in ANB we use both source and target data for training, while we aim to maximize the likelihood only with respect to the target data. Towards this goal, ANB maintains weights for instances in both source and target domains. At each iteration, the weights are increased for the target-domain data, and decreased for the source-domain data, under the assumption that the classifier continues to improve with respect to pre-

dicting target data. This behavior is controlled by a constant lambda (λ). Furthermore, for the source domain we do not use domain specific features, but only domain independent features, as only they can serve as a bridge for transferring information from source to target. As opposed to the source domain, for the target domain, we use the whole vocabulary, i.e., target-specific features and domain independent features.

In the original formulation of the ANB algorithm, the authors assume that no labeled target data is available. We modify the algorithm to allow it to make use of labeled target data, in cases where a small amount of such data is available. We follow the notation in (Tan et al., 2009) to describe the ANB algorithm:

E-step:

$$P(c_k|s_i) \propto P(c_k) \prod_{v \in V} (P(f_v|c_k))^{N_{v,i}}$$

M-step:

$$P(c_k) = \frac{(1-\lambda) * \sum_{i \in D_s} P(c_k|s_i) + \lambda * \sum_{i \in D_t} P(c_k|s_i)}{(1-\lambda) * |D_s| + \lambda * |D_t|}$$

$$P(f_v|c_k) = \frac{(1-\lambda) * \left(\eta_v^s * N_{v,k}^s\right) + \lambda * \left(N_{v,k}^t\right) + 1}{(1-\lambda) * \sum_{v=1}^{|V|} \left(\eta_v^s * N_{v,k}^s\right) + \lambda * \sum_{v=1}^{|V|} \left(N_{v,k}^t\right) + |V|}$$

In the above formulas, $N_{v,k}^s$ and $N_{v,k}^t$ denote the number of appearances of feature f_v in class c_k, for source domain (D_s) and target domain (D_t), respectively, and are obtained as follows:

$$N_{v,k}^s = \sum_{i \in D_s} N_{v,i}^s * P(c_k|s_i), N_{v,k}^t = \sum_{i \in D_t} N_{v,i}^t * P(c_k|s_i)$$

Furthermore, λ is a parameter for controlling the weights for the source domain versus target domain instances. The value of λ changes with the number of iterations (τ), which is expressed as: $\lambda = \min(\delta * \tau, 1)$ and $\tau \in \{1,2,3,...\}$. Here, δ is a constant (in our work we used $\delta = 0.2$, similar to the value used by Tan et al.); η_v^s is 0 if $f_v \notin V_{FCE}$ and 1 $f_v \in V_{FCE}$.

We use this algorithm under two scenarios. First, we assume that no labeled target data is available. In the second scenario, we assume that a small amount of labeled target data is available (in addition to labeled source and unlabeled target data). The difference between the two scenarios is captured as follows:
Case 1: During the first iteration $D_t = \phi$ in the M-step. From the second iteration onwards, D_t is D_{t_unlab}, with labels predicted by the current model, until convergence is met. This case corresponds to the original version of the algorithm (Tan et al., 2009).
Case 2: During the first iteration D_t is given by D_{t_lab} in the above M-step. From the second iteration onwards D_t consists of D_{t_lab} and D_{t_unlab} with labels predicted based on the current model, until we reach a convergence point. This case captures the modification that we made to the original algorithm.

More precisely, in the first case, we assume that the target domain has only unlabeled data. Thus, we first train a classifier using the source domain labeled data and predict the corresponding labels for the target domain unlabeled data. Starting with the second iteration, we train a classifier using both source and target data, and use the trained classifier to predict labels for the target domain unlabeled data. This process is repeated iteratively until we meet a convergence point, i.e., until we have the same labels for the target domain unlabeled instances for two consecutive iterations. During this iterative process, we use only the generalized features for the source domain, whereas for the target domain we use the whole vocabulary as features. Also, during training, we reduce the weight for the source domain instances (to decrease the influence of the source), while increasing the weight for the new target domain instances, in an effort to help predict the target domain instances accurately.

In the second case, we assume that the target domain has a small amount of labeled data and also unlabeled data. This case is similar to the first case, except that we used both source-domain labeled data along with target-domain labeled data instead of using only source labeled data to initially train the classifier.

3.2 Features

As mentioned earlier, we focus on sentence level sentiment classification and build classifiers based on *grams* derived from structured syntax trees. For a given sentence, we retrieve its complete syntax tree using the Stanford parser described in (Klein and Manning, 2003). A syntax tree is an ordered tree consisting of a root node, branch nodes and leaf nodes. For example, if the original sentence is *"Too simple for its own good."*, the syntax tree obtained using the Stanford parser is shown in Figure 1.

To generate features from syntax trees, we construct *"grams"*, which are defined as subtrees of the complete syntax trees. Specifically, we use the following classes of subtrees (grams) for our problem:
All Grams with Leaf Nodes: This representation has all the possible parent-child subtrees as features.
Unigrams with Leaf Nodes: This type representation contains unigram subtrees as features. Unigram subtrees are defined as pairs composed of a child node which is the leaf, and the corresponding parent node.
Unigrams without Leaf Nodes: This type of fea-

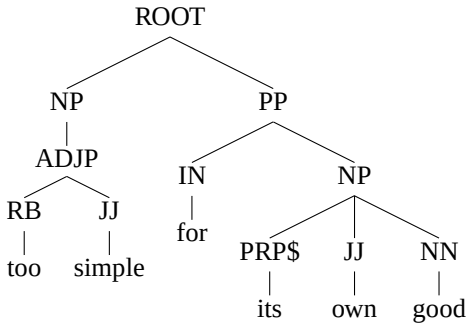

Figure 1: Syntax tree for the phrase *"too simple for its own good"* generated using the Stanford parser.

ture representation contains all possible unigram subtrees as features, except the unigrams with leaf nodes. Thus, unigram subtrees are defined as pairs composed of a child node and its corresponding parent, where the child node is not a leaf node.

All Unigrams: All possible unigrams present in the syntax tree are used as features. The combination of unigrams with leaf nodes and unigrams without leaf nodes gives all possible unigrams.

Experiments performed with these types of grams as features, using a supervised algorithm in a single domain, showed that they can effectively predict the sentiment of review sentences. However, the main challenge for performing domain adaptation is to select the domain independent features, i.e., features that bridge between the source domain and target domain. As mentioned earlier, domain independent features have the same meaning in both source and target domains. The domain independent features are important because these features occur frequently in both domains and can be used in order to transfer knowledge from source to target.

Our goal is to learn a classifier based on source domain labeled data along with target domain unlabeled data, or on source domain labeled data along with target domain labeled and unlabeled data, and use the classifier to predict the labels for target domain unlabeled instances. Source data should be represented using domain independent features, while target data is represented using all features in the target domain (including the specific features), as we want to learn to predict target well. We use the Frequently Co-occurring Entropy *(FCE)* method as described by Tan et al., (2009) to retrieve the domain independent features, also known as generalized features. This measure satisfies the following two criteria: (a) independent features occur frequently in both source and target domains; (b) Independent features must have similar occurring probability. To satisfy these requirements, we used the formula from (Tan et al., 2009):

$$f_v = \log\left(\frac{P_s(v) * P_t(v)}{P_s(v) - P_t(v)}\right)$$

where f_v represents the entropy value for the feature v, $P_s(v)$ is the probability of feature v occurring in the source domain and $P_t(v)$ is the probability of feature v occurring in the target domain. Specifically, we have:

$$P_s(v) = \frac{(N_v^s + \alpha)}{(D_s + 2 * \alpha)}, P_s(v) = \frac{(N_v^t + \alpha)}{(D_t + 2 * \alpha)}$$

where N_v^s and N_v^t denote the number of times feature v has occurred in the source domain and target domain, respectively. D_s and D_t denote the total number of instances in the source domain and target domain, respectively. We have used a constant α to smooth probabilities and avoid overflow. In our work, α value is set as 0.0001. To avoid the division by zero when both the source domain and the target domain probabilities are the same, a constant factor β is introduced, which in our work is set to 0.0001, and the above formula is modified as follows:

$$f_v = \log\left(\frac{P_s(v) * P_t(v)}{(P_s(v) - P_t(v)) + \beta}\right)$$

4 EXPERIMENTS

Before performing the domain adaptation experiments, we studied the necessity of using all the grams (within a category) as opposed to reducing the number of grams in the target domain, based on frequency. Specifically, we performed experiments where target data was represented with all grams or only with grams that occur more than one time, two times and three times, respectively. Similarly, we varied the number of FCE grams in order to identify the number of domain independent features to be used for representing the source data. Specifically, we ran experiments with 50 and 100 domain independent features. The results, reported in (Cheeti, 2012), show that it is preferable to remove grams that occur only once, and also that it is preferable to use 100 FCE as opposed to only 50. Thus, for all the experiments reported here (corresponding to various source and target combinations, along with different gram representations), we use the top 100 FCE grams as generalized features, and consider only grams that appear more than one time in source and target domains.

The purpose of these experiments is to compare the performance of sentiment classification using the Adapted Naïve Bayes algorithm (ANB), a domain adaptation classifier, and the supervised Naïve Bayes Multinomial algorithm (NBM), a domain-specific classifier, across various combinations of source and

target domains. We also compare the performance of the ANB classifier for cross-domain sentiment classification when using a small amount of target domain labeled data versus the ANB classifier without any target domain labeled data.

We used 3 domains and 4 datasets in our experiments (see Section 5 for more details). In each experiment, we start with a labeled set of target data, and we split this set into two subsets, one that will be used as target labeled data and another one (larger) that will be used as target unlabeled data (by pretending that the labels are not known, in other words, not using them in the learning process). Each subset is further split into three sub-folds in order to apply the cross-validation technique. Our choice to work only with target data for which labels are known is justified by the choice of comparisons that we perform between the cross-domain adaptation algorithms and supervised baselines, as described below. Specifically, we performed the following experiments in our work:

1. **ANB_SL_TL** denotes experiments performed in a cross-domain sentiment classification framework, using our extension of the ANB classifier, which allows the labeled data from both the source domain (SL) and the target domain (TL) to be utilized. Here, along with the source domain labeled data (SL), two folds of target domain labeled data (TL) and two folds of target domain unlabeled data (TU) are used in the training phase. The resulting model is tested on the remaining fold of the target domain unlabeled data. This procedure is repeated 3 times, in order to perform 3-fold cross-validation. The labeled data (SL) from the source domain is used in its entirety with each of the three folds of the target data.

2. **ANB_SL** represents experiments using an ANB classifier trained on labeled data that comes from the source domain (SL) and unlabeled data (TU) coming from the target domain. In this scenario, the assumption is that the target domain has no labeled instances, hence knowledge from the source domain must be leveraged. At each fold, all the labeled data from the source domain is used along with two folds of unlabeled data from the target domain to learn a model, which is then evaluated on the remaining third fold of the target unlabeled data. ANB_SL is expected to be worst than ANB_SL_TU, since labeled data is coming only from the source domain, whereas ANB_SL_TU also makes use of whatever limited (but nevertheless important) amount of labeled data the target domain may have. As before, this procedure is repeated 3 times, in order to perform 3-fold cross-validation.

3. **NBM_TL** corresponds to experiments using a supervised classifier (NBM) trained only on the target domain labeled data (TL), which is also used in ANB_SL_TL. Again, one third of the target domain unlabeled data is used for evaluating the model in a 3-fold cross validation procedure. The results of this experiment are expected to be worse than the results of the domain adaptation experiments (i.e., ANB_SL_TL and ANB_SL) because the model is trained only on the labeled data (without any unlabeled data). However, being trained on the labeled data from the target domain, NBM_TL is expected to exhibit improvements over the model learned from source labeled data, namely NBM_SL.

4. **NBM_TL_TU** denotes experiments using a supervised classifier trained on target domain labeled data (TL) along with target domain unlabeled data (TU) as training instances (the labels of the TU revealed this time). The remaining one third of target domain unlabeled data is used as testing, with labels intentionally ignored so that we can asses the quality of the model. The results of this experiment are expected to be better than the results of the domain adaptation experiments (i.e., ANB_SL_TL and ANB_SL), given that all data available is used as labeled data.

5. **NBM_SL** refers to the experiments with the supervised classifier NBM trained on source domain labeled data. The results of these experiments are expected to be worst than any results where target data is used, as the resulting classifier is tested on the target data, which is presumably different from the source.

Our experiments are designed to compare the results of the domain adaptation algorithms (ANB_SL_TL and ANB_SL) with the results of supervised domain specific algorithms, where either target data (NBM_TL and NBM_TL_TU) or source data (NBM_SL) is used. We expect the results of the NBM_SL classifier to be worse than the results of classifiers where any target data is used, unless the source data is very similar to target data. Furthermore, we expect the following relationship among the results of the classifiers that make use of any target data: NBM_TL_TU $>$ ANB_SL_TL $>$NBM_TL $>$ ANB_SL$>$ NBM_SL

5 DATASETS

In our experiments we used customer reviews from Amazon (by crawling the Amazon customer reviews) and BestBuy (for which the BestBuy API package at https://bbyopen.com/developer was used). We assumed that reviews with ratings 4 or 5 are positive and reviews with ratings 1 and 2 are negative.

We considered three domains in our study: movie reviews, DVD reviews and kitchen appliance reviews. Our goal was to include a pair of two more closely re-

lated domains (i.e., movies and DVDs), and two pairs of more distant domains (movies and kitchen appliances, and DVDs and kitchen appliances). This allows us to study the effect that the closeness of the domains has on the performance of the domain adaptation algorithms. Intuitively, we expect that the closer the domains, the more knowledge can be transferred from source to target, and thus the better the performance of the domain adaptation algorithms.

We collected an equal number of positive and negative reviews for each domain, as we did not aim to study the effect of data imbalance on the results of domain adaptation classifiers. Specifically, we collected 400 reviews (200 positive and 200 negative) for each of the movie (M), DVD (D) and kitchen appliance (K) domains. We extracted as many reviews as available from BestBuy (using the BestBuy API) and the rest were manually crawled from Amazon.

In addition to the initial 400 reviews for DVDs, we collected 400 more DVD reviews, as we wanted to study the performance of the algorithms with the size of the data available, in our case 400 versus 800 instances available. We denote the dataset containing 800 DVD reviews by D'. Using these datasets, we run experiments with the following source/target combinations in our study: $D \rightarrow M, M \rightarrow D, M \rightarrow K, K \rightarrow M, D \rightarrow K, K \rightarrow D$. Here, the left side of the arrow represents the source domain and the right side of the arrow represents the target domain.

To summarize, we assembled the datasets described above with the following questions in mind:
1. How does the distance between source and target domains affect the performance of the domain adaptation classifiers? Can we learn better classifiers for $D \rightarrow M, M \rightarrow D$ combinations as opposed to $M \rightarrow K, K \rightarrow M, D \rightarrow K, K \rightarrow D$ combinations?
2. Is there a similar amount of knowledge transfered between 2 domains, regardless of the direction? In other words, do we observe similar performance for $D \rightarrow M$ and $M \rightarrow D$?
3. How does the number of instances in the target domain affect the performance? Is $M \rightarrow D'$ better than $M \rightarrow D$ classifier?

6 EXPERIMENTAL RESULTS

As mentioned in Section 4, we only used grams that occur more than once in the target dataset (in other words, we removed grams that occurred just one time). Furthermore, we used 100 generalized features to represent the source data. We measured the performance of the classifiers using the F1 measure.

The values of the averaged F1 measure (over the positive and negative classes) are shown in Figure 2 for all grams with leaf nodes. The trend observed is consistent throughout all of our experiments: unigrams without leaf nodes, all unigrams, all grams (results not shown due to space constraints). However, the results for "all grams with leaf nodes " are better than the results for "unigrams with leaf nodes", which in turn are better than the results for "all unigrams". Finally, "unigrams without leaf nodes" give the worst results. In other words, "all grams with leaf nodes" contain more predictive information than simply unigrams. On the other hand, if unigrams are used, the most predictive ones are those with leaf nodes. Therefore, the actual words used in the sentences are important for classification. But the syntactic structure of the sentence, captured in the "all grams with leaf nodes" is also important.

Furthermore, Figure 2 shows that the F1 values for NBM_TL_TU are consistently better than the results of all the other classifiers. This is what we expected as, in this experiment, we assume that all data is labeled (labels of the "unlabeled" data used for the domain adaptation classifiers are revealed here). Also, the results of ANB_SL_TL are better than the results of ANB_SL - indeed, our modification of the original domain adaptation algorithm, where we make use of some labeled target data in addition to unlabeled target data, shows better performance. As expected, NBM_TL gives worse results than ANB_SL_TL.

The results of the comparison between NBM_SL (supervised classifier trained only on source) and ANB_SL (domain adaptation classifier that uses source labeled data and only unlabeled target data) show that ANB_SL is not always better than than NBM_SL. Again, our modification to include some target labeled data (when available) is beneficial, as the domain adaptation algorithm is not always justified otherwise - a classifier learned just from source can sometimes give better results. One possible explanation for this is that in the absence of any labeled target data, the original labels assigned to the unlabeled data are not so good, and ultimately the performance is worse than learning from source alone.

From the graph, we can also see that the transfer of knowledge is not symmetric, as the classifiers learned depend on the set of instances provided as input and the input instances could be better in a direction as opposed to another. Thus, we observed that the performance of sentiment classification is better for $M \rightarrow D, M \rightarrow K, K \rightarrow D$ as compared to $D \rightarrow M, K \rightarrow M, D \rightarrow K$.

The knowledge transfer is also influenced by the distance between domains. As D and M are more closely related than D and K, or M and K, the results

of the domain adaptation algorithms are generally better for the *D*/*M* source/target combinations. When the domains are closer related, the use of source labeled data is more helpful than in the cases where the domains are more distant. The results for the combinations *D*/*K* are better than the results for *D*/*M*. This can be explained by the fact that both *D* and *K* are product review, whereas *M* represents movie reviews.

At last, we observed that the $M \rightarrow D$ classifier results are better than $M \rightarrow D'$. One possible explanation for this is that as we increase the number of instances in the target domain, the classifier learned from the source domain may not be as informative to predict the labels for the target domain instances during the EM iterations.

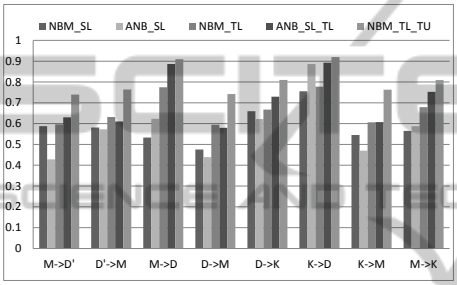

Figure 2: Domain Adaptation using All Grams with Leaf Nodes coming from Complete Syntax Trees (CST).

7 CONCLUSIONS AND FUTURE WORK

We have studied the use of a domain adaptation algorithm, specifically the ANB algorithm, for learning classifiers for predicting the sentiment of reviews across domains. In addition to using the original version of the ANB algorithm, which makes use of labeled source data and unlabeled target data, we have also used a modified version of the algorithm, which makes use of target labeled data, in addition of labeled source data and unlabeled target data.

Based on the experimental results, we can conclude that the ANB classifier increases the performance of sentiment classification across domains especially when we use some labeled target data with our modified version of the original ANB. We can also conclude that the results obtained using "all grams with leaf nodes ">"unigrams with leaf nodes ">"all unigrams">"unigrams without leaf nodes".

However, while our ANB approach performed well across different domains, there are some ideas that we would like to explore in future work. First, we would like to perform sentiment classification across domains by training a classifier using grams extracted from minimal complete trees (obtained using sentiment-based pruning strategies). The expectation here is that some parts of the tree that might not be useful for the sentiment classification problem will be removed. Second, we would like to perform sentiment classification across domains, by considering grams extracted from path subtrees (obtained using adjective-based pruning strategies). As for minimal complete trees, we expect that the results might be better when we remove parts of the trees that might not be predictive. Last, we would like to explore the identification and use of "interesting" part of speech (POS) patterns for a given set of sentences, with the expectation that more carefully designed pattern features might result in better results.

REFERENCES

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *ACL*.

Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proc. of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06. ACL.

Cheeti, S. (2012). Cross-domain sentiment classification using grams derived from syntax trees and an adapted naïve bayes approach (thesis).

Dai, W., Xue, G.-R., Yang, Q., and Yu, Y. (2007). Transferring naive bayes classifiers for text classification. In *Proc. of the 22nd national conference on Artificial intelligence - Volume 1*, AAAI'07. AAAI Press.

Harb, A., Plantié, M., Dray, G., Roche, M., Trousset, F., and Poncelet, P. (2008). Web opinion mining: how to extract opinions from blogs? In *Proc. of the 5th international conference on Soft computing as transdisciplinary science and technology*, CSTST '08. ACM.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proc. of the 41st Meeting of the ACL*.

Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. (1998). Learning to classify text from labeled and unlabeled documents. In *Proc. of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98. AAAI.

Pan, S. J., Ni, X., Sun, J.-T., Yang, Q., and Chen, Z. (2010). Cross-domain sentiment classification via spectral feature alignment. In *Proc. of the 19th international conference on World wide web*, WWW '10. ACM.

Tan, S., Cheng, X., Wang, Y., and Xu, H. (2009). Adapting naive bayes to domain adaptation for sentiment analysis. *Advances In Information Retrieval Proceedings*, 5478.

Zhang, W., Li, P., and Zhu, Q. (2010). Sentiment classification based on syntax tree pruning and tree kernel. In *Proc. of the 2010 Seventh Web Information Systems and Applications Conference*, WISA '10. IEEE Computer Society.