CROSS-DOMAIN SENTIMENT CLASSIFICATION USING GRAMS DERIVED FROM SYNTAX TREES AND AN ADAPTED NAIVE BAYES APPROACH

by

SRILAXMI CHEETI

B.Tech, Jawaharlal Nehru Technology University (JNTU), India, 2008

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2012

Approved by:

Major Professor
Doina Caragea

Abstract

There is an increasing amount of user-generated information in online documents, including user opinions on various topics and products such as movies, DVDs, kitchen appliances, etc. To make use of such opinions, it is useful to identify the polarity of the opinion, in other words, to perform sentiment classification. The goal of sentiment classification is to classify a given text/document as either positive, negative or neutral based on the words present in the document. Supervised learning approaches have been successfully used for sentiment classification in domains that are rich in labeled data. Some of these approaches make use of features such as unigrams, bigrams, sentiment words, adjective words, syntax trees (or variations of trees obtained using pruning strategies), etc. However, for some domains the amount of labeled data can be relatively small and we cannot train an accurate classifier using the supervised learning approach. Therefore, it is useful to study domain adaptation techniques that can transfer knowledge from a source domain that has labeled data to a target domain that has little or no labeled data, but a large amount of unlabeled data. We address this problem in the context of product reviews, specifically reviews of movies, DVDs and kitchen appliances. Our approach uses an *Adapted Naïve Bayes classifier* (ANB) on top of the Expectation Maximization (EM) algorithm to predict the sentiment of a sentence. We use grams derived from complete syntax trees or from syntax subtrees as features, when training the ANB classifier. More precisely, we extract grams from syntax trees corresponding to sentences in either the source or target domains. To be able to transfer knowledge from source to target, we identify generalized features (grams) using the frequently co-occurring entropy (FCE) method, and represent the source instances using these generalized features. The target instances are represented with all grams occurring in the target, or with a reduced grams set obtained by removing infrequent grams. We experiment with different types of

grams in a supervised framework in order to identify the most predictive types of gram, and further use those grams in the domain adaptation framework. Experimental results on several cross-domains task show that domain adaptation approaches that combine source and target data (small amount of labeled and some unlabeled data) can help learn classifiers for the target that are better than those learned from the labeled target data alone.

Table of Contents

List of Figures

List of Tables

Acknowledgments

While this thesis is my own work, it benefited from the insights and direction of several people. I would like to thank them all for their help and support.

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Doina Caragea, for her excellent guidance, caring, patience, and for providing me with an excellent atmosphere for doing research. Without her guidance and persistent help this thesis would not have been possible. I would like to thank her for all the valuable discussions and inputs she provided during the last two years. Her patience and support helped me overcome many crisis situations and finish this thesis. Throughout my studies at KSU, she provided encouragement, sound advice, good teaching, good company, and lots of great ideas. I would have been lost without her.

I would like to thank Dr. Amtoft Torben for being a member of my M.S. committee, and for educating me about some of the important concepts in algorithms and about ways to tackle some of the hardest problems.

I would also like to thank Dr. Mitchell L. Neilsen, for being a member of my M.S. committee. His classes are a great source of information and motivate students to think of simple solutions for some of the most challenging problems in real time operating systems.

I would like to thank my family members, Mr. Cheeti Hanmantha Rao, Mrs. Cheeti Rama Devi and my brother Mr. Cheeti Srinivas for their love and support at every stage of my life. It is because of their support and encouragement that I am able to complete my Masters. I would also like thank my friends and colleagues especially Sandeep Solanki, Karthik Tangirala, Ana Stanescu, Nic Herndon for helping me in the early days of my M.S. studies and for valuable discussions.

Finally, I would like to thank my husband, Kalyana Koka. He was always there cheering me up and stood by me through the good and bad times.

Chapter 1

Introduction

In this chapter, we will first provide some motivation for the sentiment analysis problem in Section 1.1. We state the problem addressed and emphasize challenges of this problem in Section 1.2. Finally, we give an outline of the proposed approaches in Section 1.3.

1.1 Motivation for Sentiment Classification

The amount of information available on the web is increasing tremendously every day. Some of this information is provided by internet users in the form of reviews, blogs, webpages, etc., and can be useful for other users or for companies targeting internet users. For example, product reviews contain information that can be helpful in the decision making process of new customers looking for various products. Assuming that several companies such as Sony, Canon, Nikon make the same product (e.g., camera), a customer might be interested in buying the best camera available, within a particular price range, regardless of the producing company. In order to pick the best camera, that customer needs to know what are the pros and cons of the cameras made by different companies. In other words, the customer needs to classify the online information (i.e., the reviews) as positive, negative or neutral.

Similarly, customer reviews can be useful for the companies manufacturing products, as they can learn about customer's likes and dislikes and adjust the products accordingly, or use that information to train recommender systems to recommend products to users. For example, companies like Amazon, Motorola, ATT, Verizon can identify reviews and classify

them as positive, negative or neutral. This information will help companies to come up with ideas for new features that the customers are looking for, which, in turn, can result in an increase in the revenue for the company. Furthermore, companies can also use the information that is available online to provide recommendations related to products, movies, restaurants, maps, good community schools, etc. based on the likes and dislikes of a person.

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 greatly needed, 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 this context, a 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, including NLP features.

1.2 Problem Addressed and Challenges

Sentiment classification, in general, is a broad problem, which can be addressed at various levels. For example, we can talk about sentiment classification at word level, sentence level 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 sentence level. Sentence level classification is more difficult 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 (and for that matter 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*. We aim to use machine learning approaches to address this problem. Generally, with enough training data this approach is feasible and can result in accurate domain specific classifiers. For example, we can use movie review data to learn a movie sentiment classifier and use it to predict the sentiment of new movie reviews. However, in real world applications, 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 can be used on a target domain, but are learned from both target and another source domain. 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, excellent* express positive sentiment in both kitchen appliance domain, as well as camera domain. Words such as *bad, worse* express negative sentiment across both kitchen appliance and camera domains. However, words such as *tasteful, tasteless* express sentiments in kitchen appliance domain and may or may not express any sentiment in the camera domain. Words such as *lens, sleek, megapixel* express sentiment in camera domain and may or may not express any sentiment in the kitchen appliance domain. In cross-domain classification problems, the general goal is to leverage labeled data in the source domain and, possibly, some labeled data in the target domain, together with unlabeled data from the target. Under this scenario, *we aim to learn cross-domain classifiers (at sentence level) for predicting the sentiment of target instances by using data available in both source and target domains*.

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 a lot of 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 previous approaches [Pan et al., 2010], [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 our work, we used 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 as assumed to be generalized features and used to represent the source domain. Furthermore, to combine source and target data, we use an EM based naïve Bayes classifier proposed also by Tan et al. [2009]. In this approach, 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 classifier can be used for predicting the target domain instances. Originally, the approach in [Tan et al., 2009] assumes labeled source data and unlabeled target data. In our implementation, we can also use labeled target domain data.

1.3 High Level Overview of Proposed Approaches

Our goal is to use features extracted from subtrees of a complete syntax tree or a structured syntax tree to learn machine learning classifiers for predicting the sentiment of customer product reviews in a cross-domain scenario. To learn classifiers, we use an adapted naïve Bayes algorithm (ANB) built on top of the expectation maximization (EM) algorithm.

In the initial phase of the work, we experimented with complete syntax trees, subtrees of complete syntax trees or path trees as features, in a domain specific scenario, to learn about the predictiveness of such features with respect to the sentiment classification problem. Specifically, we ran several experiments with different kinds of trees as features using supervised machine learning algorithms such as SVM and naïve Bayes. Given the promising results of the syntax trees in a supervised learning framework, we decided to use them also in the domain adaptation framework (i.e., with ANB classifier). As mentioned in the previous section, we used frequently co-occurring entropy method (FCE, as described in [Tan et al., 2009]) to identify generalized features. This method calculates entropy values for features extracted from syntax trees (by comparing their occurrence frequency in target versus source), and ranks them in decreasing order of the entropy values. The top 50 or 100 FCE features are considered to be domain independent features, and are used to represent instances in the source domain.

We used ANB under the following two scenarios:

1. **Case 1: Assume that the source domain contains labeled data and the target domain has only unlabeled data.**

   Here, we first train a classifier using the source domain labeled data and predict the corresponding labels for the target domain unlabeled data. From the second iteration onwards, we train a combined classifier based on the labels predicted in the previous iteration for target domain unlabeled data and source domain labeled data. We use the trained classifier to predict labels for the target domain unlabeled instances. This process is repeated iteratively until we meet a convergence point where we have the same labels for the target domain unlabeled instances for the 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. We perform a 3 fold cross validation on target domain data. More precisely, from the second iteration onwards, we consider 2 folds of the target domain (unlabeled data)

along with source domain (labeled data) as our training data and use the remaining one fold of the target domain unlabeled data as test data. During the iterative process, we reduce the weight for the source domain instances (thus, decreasing the influence of the source on the final classifier), while increasing the weight for the new target domain instances, in an effort to help predict the target domain instances accurately.

2. **Case 2: Assume that the source domain contains labeled data and the target domain has small amount of labeled data and unlabeled data.**

   This second 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 domain labeled data as described in Case 1. In Case 2, we perform 3 fold cross validation on target domain as well. Here, from the second iteration onwards, we consider 2 folds of the target domain (labeled and unlabeled data) along with source domain (labeled data) as our training data and use the remaining one fold of the target domain unlabeled data as test data.

The rest of the thesis is organized as follows: Chapter 2 describes the previous work on cross-domain sentiment classification. In Chapter 3, we formulate the problem of sentiment classification and then explain the various approaches that we used in our work, along with some detailed examples. Chapter 4 explains the dataset, experimental setup discussing the various experiments that we have performed and also the research questions that we have addressed. In Chapter 5, we discuss the results of the experiments and explain the usefulness of our proposed approaches for the cross-domain sentiment classification problem. Finally, in Chapter 6 we conclude our work and present directions for future work in Chapter 7.

Chapter 2

Related Work

This chapter gives detailed information about the previous work on sentiment classification across domains. The information available on the web is growing tremendously day by day. Sentiment classification across domains is very challenging because generally, a classifier trained on one domain cannot predict the instances from a different domain appropriately. This is because domain specific features have different meanings in different domains. The biggest challenge involved in performing sentiment classification experiments depends on selecting features and Machine Learning algorithms to use for different datasets. In this chapter, we will give an overview of various types of features, Machine Learning algorithms, the datasets that were used by various authors with their corresponding results and also the type of classification (either sentence level or document level) that they addressed.

Li and Zong [2008] proposed two approaches for cross-domain sentiment classification. One is the feature level fusion and the other is the classifier level fusion approach. In the feature level fusion approach first, the authors constructed feature sets (f1,f2,f3.....) individually in different domains using the training data. Next, Li and Zong [2008] combined all the individual feature sets from different domains into a single feature set (F) and used it to train a classifier. Finally, the authors used this classifier in order to predict the instances from different domains. The disadvantage using this method is that they cannot assign different weights when trying to classify the instances from different domains. For example, if we classify instances from the DVDs domain, then we cannot easily assign higher weights

to the movie domain and lower weights to the kitchen appliances domain.

In classifier level fusion method, first the authors divided the experimental data into training data (70%), development data (20%) and testing (10%) data. Next, a base classifier is learned individually in different domains using the training data. This approach combines the base classifiers and learns a meta-classifier by applying different methods such as MetaLearning method. During this approach a meta-classifier is trained for each domain, using the development data combined with the output attributes of the base classifiers as input. For example, if we want to learn a meta-classifier for the i^{th} domain, then we use the development data from the i^{th} domain along with all the output features from all the base classifiers in different domains. Next, to test the data from a particular domain, we use the meta-classifier available from the same domain. MetaLearning will automatically learn the unbalanced information (i.e, assigning higher weights to closely related domains and lower weights to more distinct domains) overcoming the disadvantage from the feature level fusion method. In the experiments performed in this work [Li and Zong, 2008], the datasets correspond to four domains: books, DVDs, electronics, and kitchen appliances. The features used are 1gram (unigram), 2gram (bi-gram), 1+2gram (unigrams and bi-grams with high bi-normal separation scores (BNS) [Forman, 2003]) and 1gram + 2gram (unigrams+bigrams). BNS is a new metric defined as $F^{-1}(tpr) - F^{-1}(fpr)$, where F^{-1} is the inverse cumulative probability distribution of the standard Normal distribution, tpr is the true positive rate and fpr is the false positive rate. The results show that the classifier level fusion performed better than the feature level fusion because it was able to capture the unbalanced information between different domains. Li and Zong [2008] have also suggested that 1gram + 2gram features are better than other types of features that they have used in their experiments.

Harb et al. [2008] introduced the AMOD (Automatic Mining of Opinion Dictionaries) approach consisting of the following three phases. The first phase, known as 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 Adjective Extraction Phase, extracts a list of adjective words with positive and negative opinions. The third phase, known as Classification Phase is used to classify the given documents using the adjective list of 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. The training dataset was retrieved from (http://www.google.com/blogsearch) using a list of seed words for the cinema domain and the test set used was the movie review data from the Natural Language Processing (NLP) Group, Cornell University (http://www.cs.cornell.edu/people/pabo/movie-reviewdata/). Harb et al. [2008] have also experimented with data from the car domain. The result shows that AMOD approach was able to classify the given documents by using a list of adjective words in a single domain.

Blitzer et al. [2007] introduced another domain adaptation strategy, which is an extension of an approach previously proposed by the same authors, called structural correspondence learning (SCL) [Blitzer et al., 2006]. This algorithm reduces the relative error due to adaptation between domains and also identifies a measure of domain similarity when compared to the original SCL. Here, the authors first choose a set of features that occur frequently in both source and target domains, also known as pivot features. Next, linear predictors are used to find the correlations between the pivot elements and all other features in the unlabeled data from both source and target domains. The performance of this algorithm depends on the selection of the pivot features. The pivot features should be good predictors of source domain labeled data. It is very important which features to consider as pivot features because they should be helpful to predict the target unlabeled instances based on the classifier learned from both the source and target domains. The pivot features are the target features that have the highest mutual information (MI) to the source domain label. The dataset used is an Amazon product reviews dataset and consists of four different products: books, DVDs, electronics and kitchen appliances. The authors assume that source domain dataset contains labeled and unlabeled data, whereas target domain dataset

contains only unlabeled data. They observe that choosing the pivot features using MI has reduced the relative error by 36%. Furthermore, when introducing 50 labeled instances from the target domain, the observed 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.

Pan et al. [2010] proposed Spectral Feature Alignment (SFA) algorithm for cross-domain sentiment classification. The process of selecting the pivot features is same as described by Blitzer et al. [2007]. However, the gap between the domains is reduced by constructing a bipartite graph and by adapting the spectral clustering techniques. The experimental dataset consists of Blitzer, Amazon, Yelp and City-search. Blitzer dataset [Blitzer et al., 2007] contains reviews for four product domains: books, DVDs, electronics and kitchen appliances. The dataset collected from Amazon, Yelp and City-search consists of three product domains: video games, electronics and software. In [Pan et al., 2010], sentiment classification is performed at the document level. The features used are the words from different domains. Pan et al. [2010], compared SFA with NoTransf (where a classifier is learned using source domain instances as training data), SCL [Blitzer et al., 2006], LSA [Deerwester et al., 1990] and FALSA (a classifier is trained based on the features learned by applying LSA on the co-occurrence matrix of domain-independent and domain-specific features and used as one of the baselines). They found that SFA [Pan et al., 2010] results are better than those of the other algorithms.

Glorot [2011] introduced a Deep Learning approach to extract the high level features from reviews in the unlabeled data when performing sentiment classification across domains. The Deep Learning algorithm aims to find out the intermediate concepts between the source and target domains. The intermediate concepts might include concepts like product quality, product price, customer service, etc. Here, the authors follow a two step process to perform sentiment classification across different domains. In the first step, they extract the high level features using a Stacked De-noising Auto-encoder (SDA) with rectifier units. The

SDA is learned in a greedy layer-wise fashion using stochastic gradient descent. In the second step, Glorot [2011] proposes to learn a classifier on the transformed labeled data from the source domain. The features used are unigrams and bigrams. The dataset used is an Amazon dataset with 26 domains. Glorot [2011] found that the Deep Learning approach results are better than the SCL [Blitzer et al., 2006] and SFA [Pan et al., 2010] results.

Zhang et al. [2010] proposed to use different kinds 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 used for single domain sentiment classification and were found to be very useful for the classification tasks considered.

Tan et al. [2009] proposed an adapted naïve Bayes (ANB) algorithm to perform cross-domain sentiment classification. The first step is to find generalized features in order to build a bridge between the source and the target domains. In order to retrieve the generalized features they used a frequently co-occurring entropy (FCE) method and picked the features with the highest entropy values as the generalized features. Next, two classifiers are learned, one from the source domain using only the generalized features from the source domain and the other from the target domain using all the features from the target domain. Then, 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 a convergence point is met. The study used Chinese domain-specific datasets: Education Reviews (Edu, from `http://blog.sohu.com/learning/`), Stock Reviews (Sto, from `http://blog.sohu.com/stock/`) and Computer Reviews (Comp, from `http://detail.zol.com.cn/`). The ANB algorithm is compared with Naïve Bayes (supervised baseline), EM-based Naïve Bayes (semi-supervised baseline), Naïve Bayes Transfer Classifier (transfer-learning baseline) and the results show that ANB performs much better than the other algorithms.

As explained earlier, our goal is to perform sentence level sentiment classification across domains. From the above mentioned previous works, we came to know that features used

with classifiers play a vital role in classification tasks. We can select either unigrams, bigrams, 1+2Gram, 1Gram+2Gram, adjective words, sentiment words or structured syntax trees as features for our classification tasks. In our work, we use different kinds of syntax subtrees as features as discussed in detail in [Zhang et al., 2010].

We have seen that there are many different kinds of algorithms such as feature level fusion, classifier level fusion [Li and Zong, 2008], SCL [Blitzer et al., 2006], AMOD [Harb et al., 2008], SFA [Pan et al., 2010], Deep Learning [Glorot, 2011], ANB [Tan et al., 2009] that can be applied to the cross-domain classification problem. Previous work (such as Pan et al. [2010], Blitzer et al. [2006]) suggested that the generalized features can be selected by using MI, FCE methods, among others. We have used FCE and ANB as discussed in [Tan et al., 2009] to perform cross-domain sentiment classification in our work. We have extended ANB by including some labeled data from the target domain. Another contribution of this work is the use of different kinds of syntax subtrees as features. The dataset that we use is manually extracted from BestBuy (`https://bbyopen.com/`) and Amazon (`http://www.amazon.com/`). The manually extracted dataset contains reviews for customer products such as DVDs, movies and kitchen appliances. In our work, we have overcome the disadvantage of having a predefined set of domain specific and domain independent words by using a FCE [Tan et al., 2009] method.

Chapter 3

Problem Definition and Approaches

This chapter describes the task of sentence level sentiment classification across domains. The chapter is organized as follows: In Section 3.1, we begin by describing some of the terms that we use in our classification problem. In Section 3.2, we give a problem definition with the help of some examples. In Section 3.3, we describe the feature representations and explain how we use them to train classifiers for the problem stated. In Section 3.6, we describe the transfer learning approaches we use for our problem. We also give a detailed description of the approaches, such as pruning strategies, Support Vector Machine (SVM) and Adapted Naïve Bayes (ANB) algorithms, that we have use in this work.

3.1 Basic Terminology

Before we describe the problem addressed, we define the basic terminology used in this work:
Domain: A domain is a class consisting of different entities. For example, products such as books, DVDs, electronics, movies, cameras are considered to be different domains.
Sentiment Classification: Customers write reviews for various products describing pros and cons, also known as sentiments. These reviews can be in the form of a sentence, a paragraph or a document. From a classifier point of view, the reviews are represented in the form of a bag of words, such as $w_1, w_2, w_3, \cdots, w_n$, where $w_i \in \text{Vocabulary}(V)$. In order to recommend products to a new customer, we need to know whether the reviews written by the old customers are recommending the product or not. We classify these reviews either

as positive or negative based on words that were used in these reviews. Thus, in this work we deal with a binary classification problem where a given review can belong to either a positive or a negative class. For any given domain, we denote the total set of sentences in the given domain by S, where $s_1, s_2, s_3, \cdots, s_m \in S$. For any given sentence s_k, where $k \in \{1, 2, 3, \cdots, m\}$, s_k can belong to either a positive or negative class denoted by y_k based on the words present in it. In our work, $y_k = 1$ if the polarity of a given sentence is positive and $y_k = -1$ if the polarity of a given sentence is negative.

Domain Adaptation: Domain adaptation, a.k.a. cross-domain learning or transfer learning, is used in many areas such as text classification, spam filtering and bioinformatics. The aim of domain adaptation is to learn a model using labeled data from a source domain combined with target domain unlabeled data and, in some cases, a small amount of target labeled data, and to predict the labels for the unlabeled instances from the target domain. In simple terms, we use the information from the source domain together with some information from the target domain to predict the unlabeled instances in the target domain. Thus, domain adaptation can leverage the knowledge learned from a source domain to the target domain. If the source and the target domains are very close, then the transfer learning is easier; otherwise transfer learning is harder and it may or may not be successful.

3.2 Problem Definition

We denote by D_s the instance distribution over the source domain, and by D_t the instance distribution over the target domain. We assume that there are M labeled instances in D_s and N instances in D_t. We explore two scenarios, one where the target domain contains only unlabeled instances and the other when the target domain contains both labeled and unlabeled instances. Our goal is to learn a classifier that has minimum error with respect to the target domain. In simple words, we try to classify the unlabeled target domain instances as either positive or negative using the combined model learned from both the source domain and the target domain instances. Our instances are review sentences. We

consider several possible combinations of source and target domains including movie, DVDs and kitchen appliances domains.

3.3 Structured Syntax Trees

As mentioned earlier, we focus on sentence level sentiment classification and build classifiers based on 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. Branch or interior nodes are labeled using non-terminals such as S, NP, JJ, as described at http://en.wikipedia.org/wiki/Parse_tree, whereas leaf nodes are labeled using terminals such as alphabets, numbers, whitespace, special characters. After retrieving the syntax tree, we apply several pruning strategies to extract subtrees from the complete syntax tree, known as structured features. For pruning, we have used a list of sentiment words available at http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html and a list of adjectives available at http://www.enchantedlearning.com/wordlist/adjectives.shtml to extract the subtrees called Minimal Complete Tees (MCT) and Path Trees (PT). These syntax subtrees are used to represent instances and provided to a supervised SVM classifier. The SVM-Light-TK tool described in [Joachims, 2002] is used for this purpose, as it can handle tree kernels (and thus avoids the need to generate linear tree features explicitly).

Given the following sentence: *"at about 95 minutes, treasure planet maintains a brisk pace as it races through the familiar story."* - the syntax tree obtained using the Stanford parser is shown in Figure 3.1.

3.3.1 MCT and PT Using Sentiment based Pruning Strategies

First, we will remove the occurrences of (, ,) or (. .) from the syntax tree generated using the Stanford parser. Next, we check whether a sentence has any sentiment words from the list used by Narayanan et al. [2009] available at http://www.cs.uic.edu/~liub/FBS/

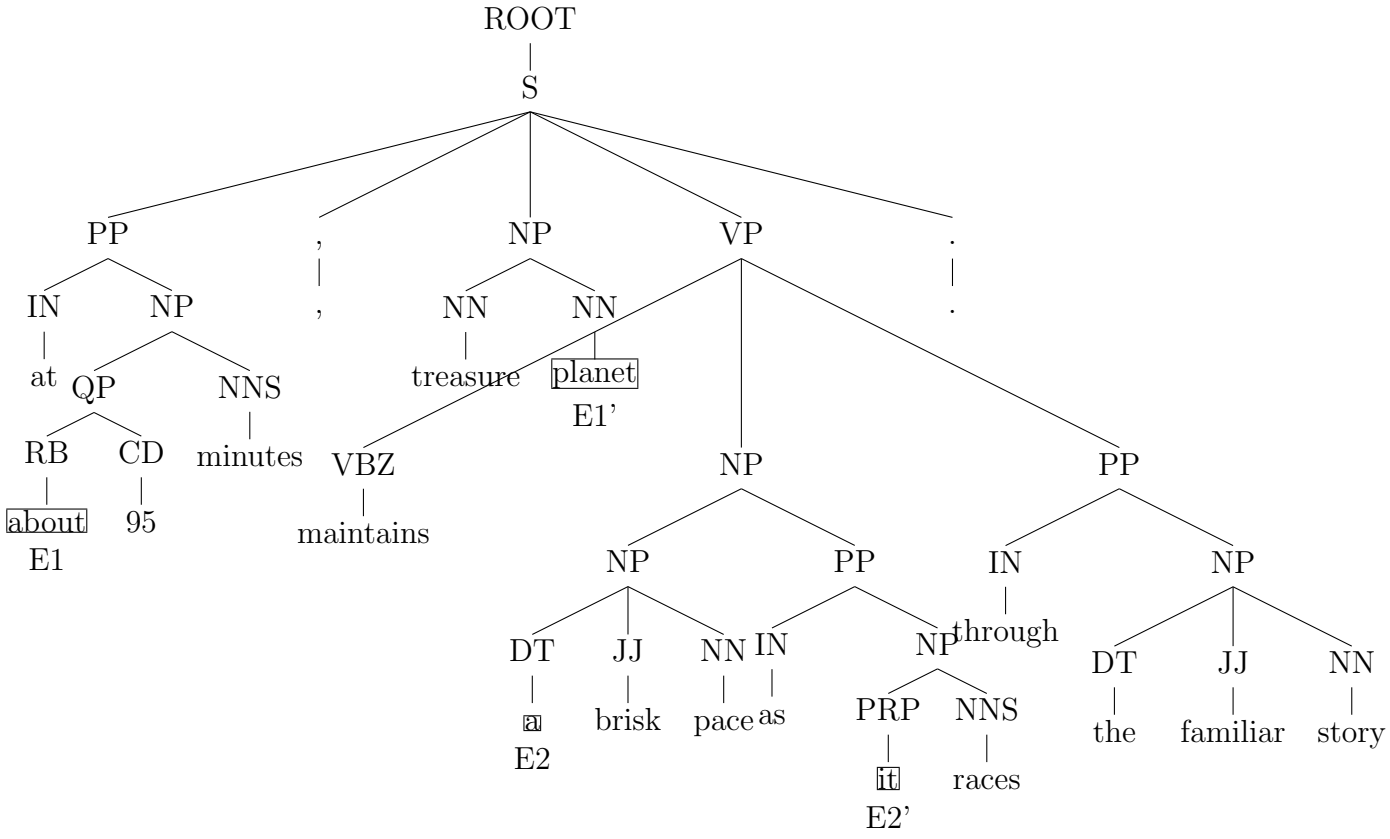

Figure 3.1: *Syntax tree generated using Stanford parser*

sentiment-analysis.html. The sentiment words for the above sentence are "treasure" and "brisk". For each sentiment word, we will define two nodes E1 and E2, using a window size of 3, where E1 is marked to the left side and E2 is marked to the right side with respect to the given sentiment word "treasure". For the sentiment word "treasure", the E1 node is "about" and the E2 node is "a", as shown in Figure 3.1, and for the sentiment word "brisk", the E1 node is "planet" and the E2 node is "it" (shown in Figure 3.1 as $E1^{'}$ and $E2^{'}$). Then, we look at a common parent of E1 and E2 and retrieve the subtree from that common parent, also known as Minimum Complete Tree (MCT) for a given sentiment word. This process is repeated for all the sentiment words in a given sentence. Then, we need to combine the MCT of all the sentiment words by checking for a common node between all the MCT in a given sentence and this resulting subtree is the final version of MCT that we will be using in our work - this tree is also called a combined MCT and reduced some noise present in the complete syntax tree.

But to eliminate further noise from MCT, we apply another pruning strategy to each MCT. During this stage, we prune all the leaf nodes along with their parent nodes that are present to the left side of the node E1, until we hit a common parent between the leaf node and the node E1. Then, we prune all the extra leaf nodes along with their parent nodes that are present to the right side of the node E2, until we hit a common parent between the leaf node and the node E2. The resulting subtree is known as a Path Tree (PT). This process of pruning is repeated for all the MCT trees present in a sentence and the final version of the PT tree is retrieved by finding a common node between all the PT trees available in a given sentence. The MCT for the sentiment word "treasure" is shown in Figure 3.2 and the PT for the sentiment word "treasure" is shown in Figure 3.3. The MCT for the sentiment word "brisk" is shown in Figure 3.4 and the PT for the sentiment word "brisk" is shown in Figure 3.5. The combined MCT that we will use in our work for the above sentence is shown in Figure 3.6 and the combined PT is shown in Figure 3.7:

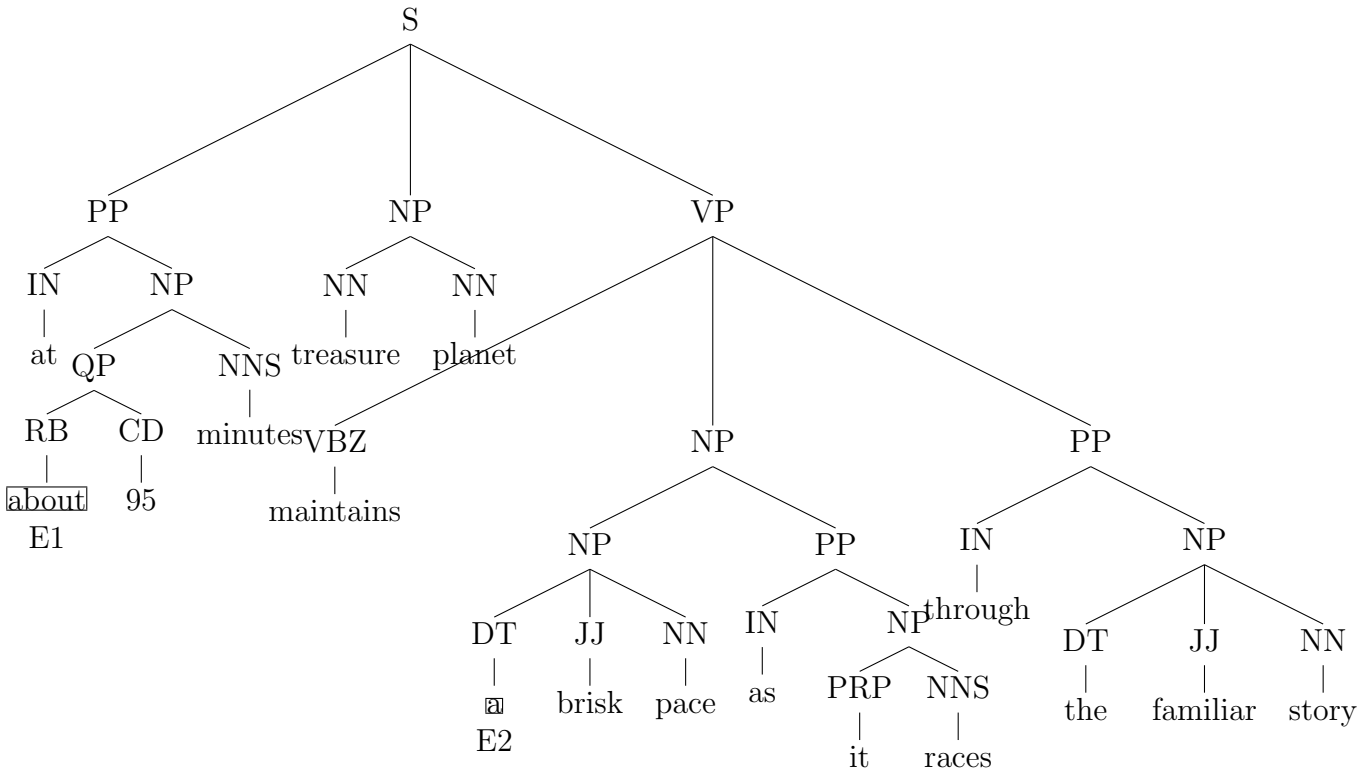

Figure 3.2: *MCT for sentiment word "treasure"*

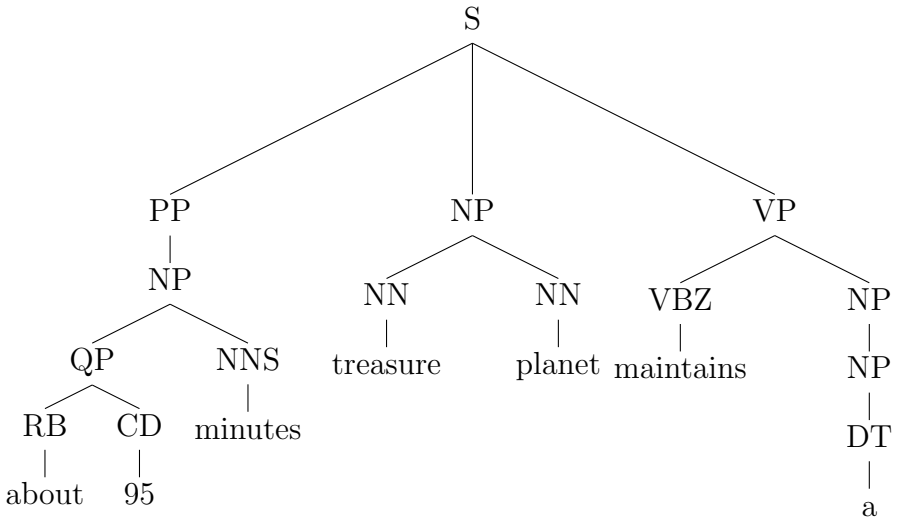

Figure 3.3: *PT for sentiment word "treasure"*

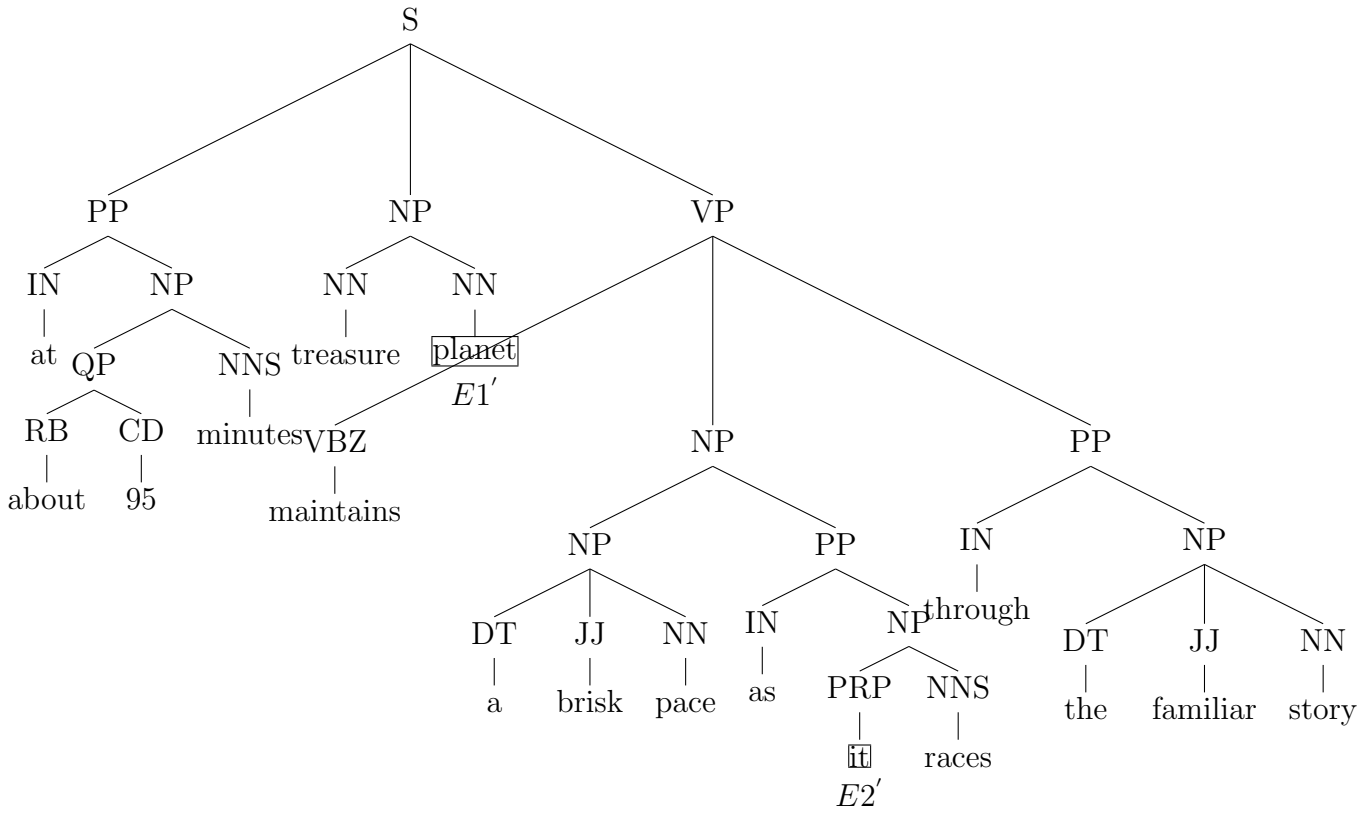

Figure 3.4: *MCT for sentiment word "brisk"*

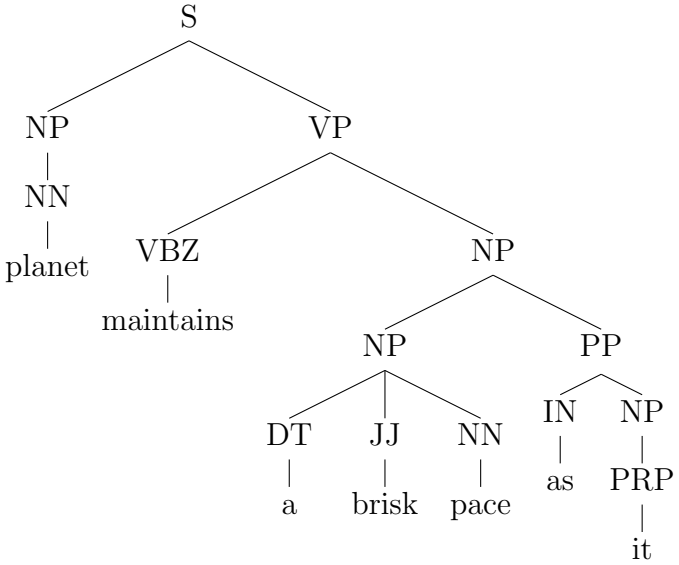

Figure 3.5: *PT for sentiment word "brisk"*

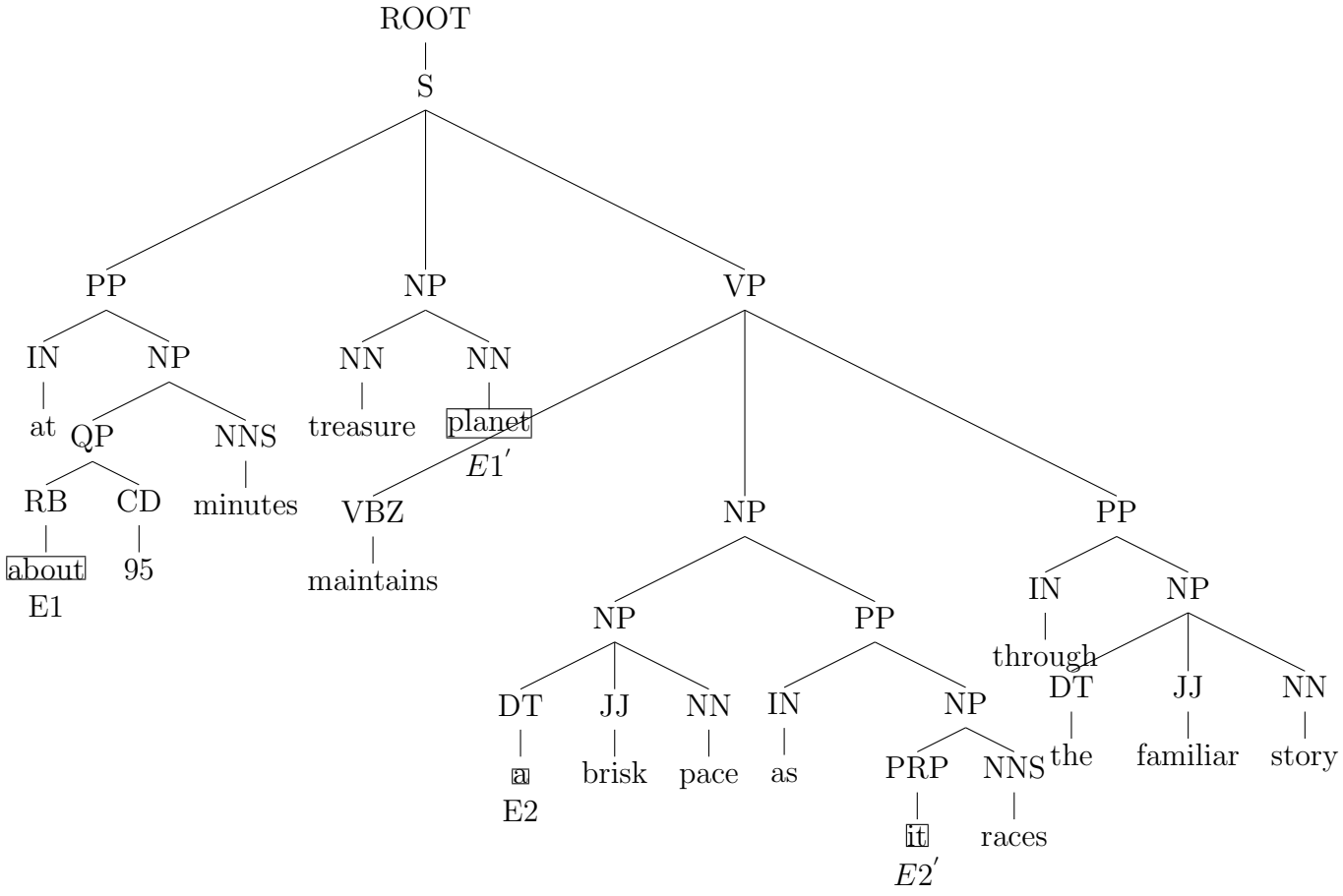

Figure 3.6: *MCT or combined tree using sentiment based pruning strategy*

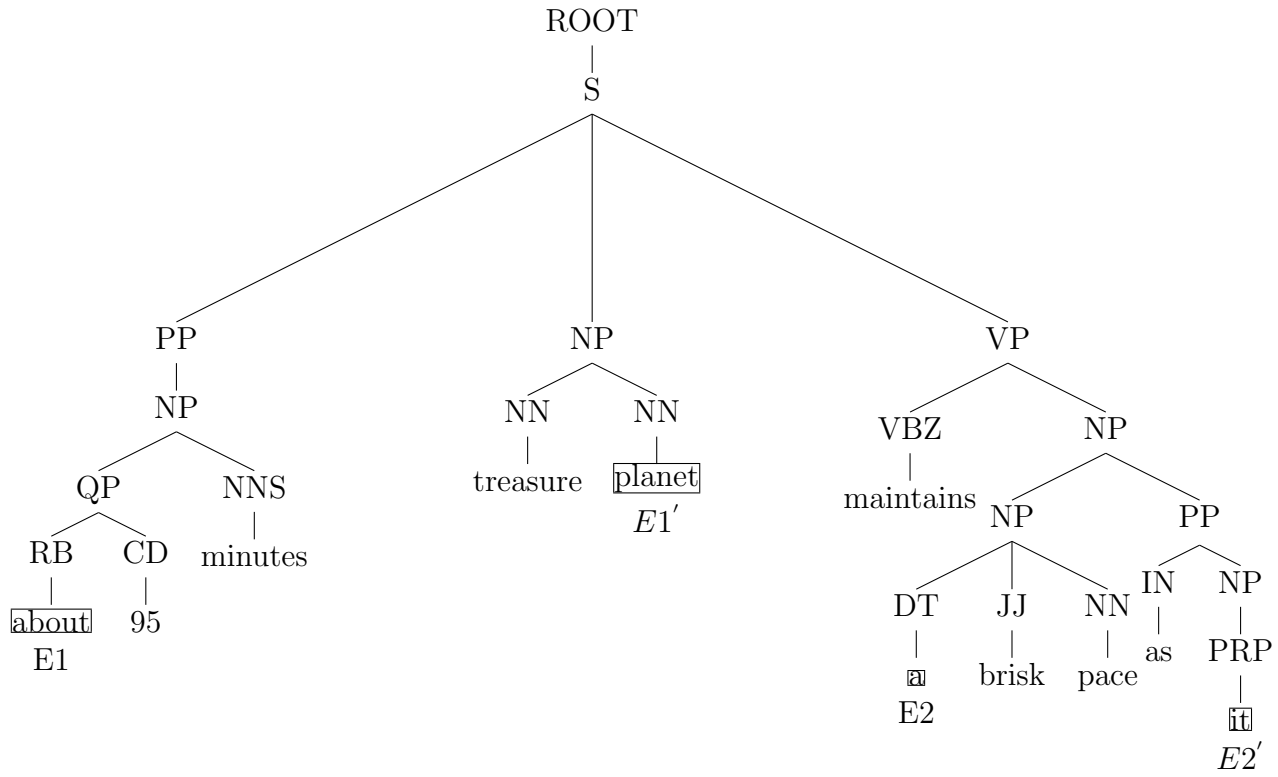

Figure 3.7: *PT or combined tree using sentiment based pruning strategy*

3.3.2 MCT and PT Using Adjective based Pruning Strategy

The adjective based pruning strategy is similar to the sentiment word based pruning strategy described in Section 3.3.1, except that in this approach we look for adjective words instead of sentiment words. First, we will check whether a sentence has any adjective words from the list at http://www.enchantedlearning.com/wordlist/adjectives.shtml. Let's assume the adjective words for the above sentence are "brisk" and "familiar", then the complete syntax tree is shown in Figure 3.8. For the adjective word "brisk", the E1 node is "planet" and the E2 node is "it" as shown in Figure 3.8. Next, for the adjective word "familiar", the E1 node is "races" and the E2 node is "story", as shown in Figure 3.9. Next, the process of extracting MCT, PT, combined MCT and combined PT is exactly same as described in Section 3.3.1. MCT for adjective word "brisk" is shown in Figure 3.10 and PT for adjective word "brisk" is shown in Figure 3.11. MCT for adjective word "familiar" is shown in Figure 3.12 and PT for adjective word "familiar" is shown in Figure 3.13. The combined MCT for the above sentence using adjective based pruning strategy is shown in Figure 3.14 and combined PT is shown in Figure 3.15:

3.4 Gram Features based on Syntax Trees

Generally, Machine Learning (ML) [Mitchell, 1997] algorithms use feature based representations for instances, where each instance is represented using a collection of features $f_1, f_2, \cdots, f_n$. In addition to using structured syntax trees with tree kernels in SVM, we also use gram features extracted from syntax trees, described in what follows.

Grams are subtrees based on either the complete syntax trees or structured trees generated using pruning strategies. If the original sentence is: *"too simple for its own good"* - the syntax tree for this sentence is represented in Figure 3.16. The sentiment word present in our given sentence is "good". The structured PT syntax tree using sentiment word based pruning strategy is shown in Figure 3.17. The adjective words are identified as "simple" and "good". The structured PT syntax tree using adjective based pruning strategy is shown in

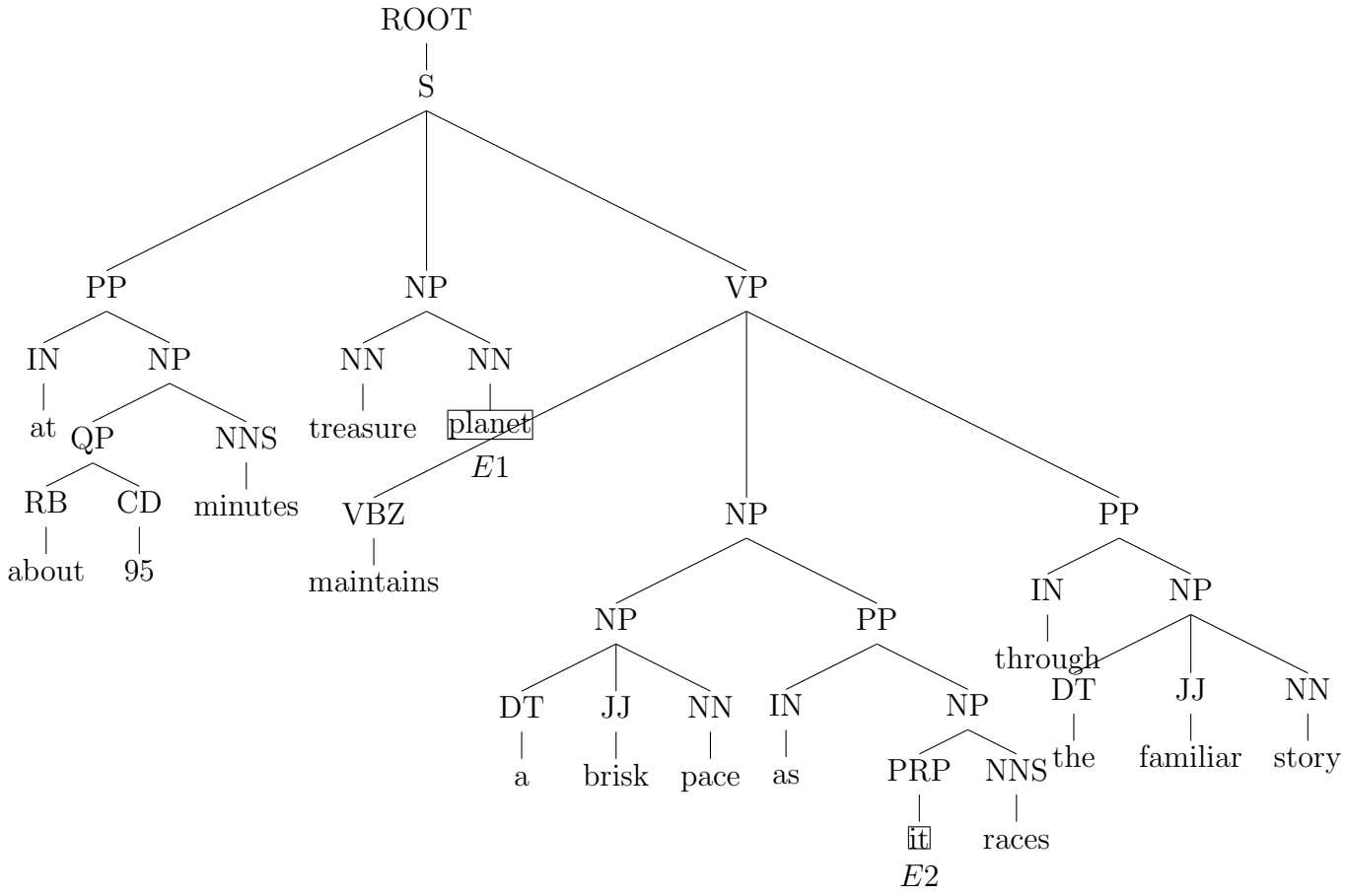

Figure 3.8: *E1 and E2 nodes for adjective word "brisk"*

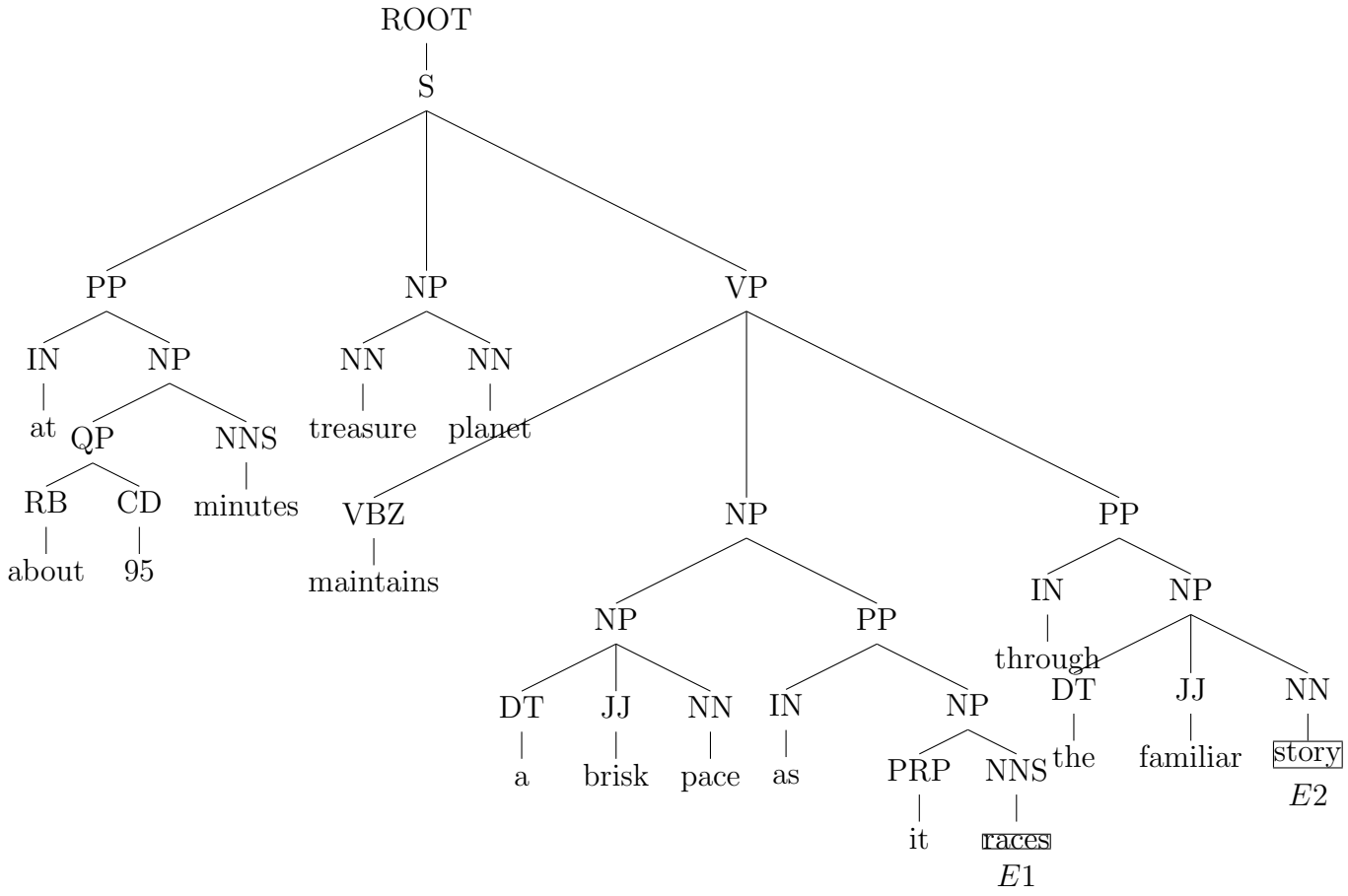

Figure 3.9: *E1 and E2 nodes for adjective word "familiar"*

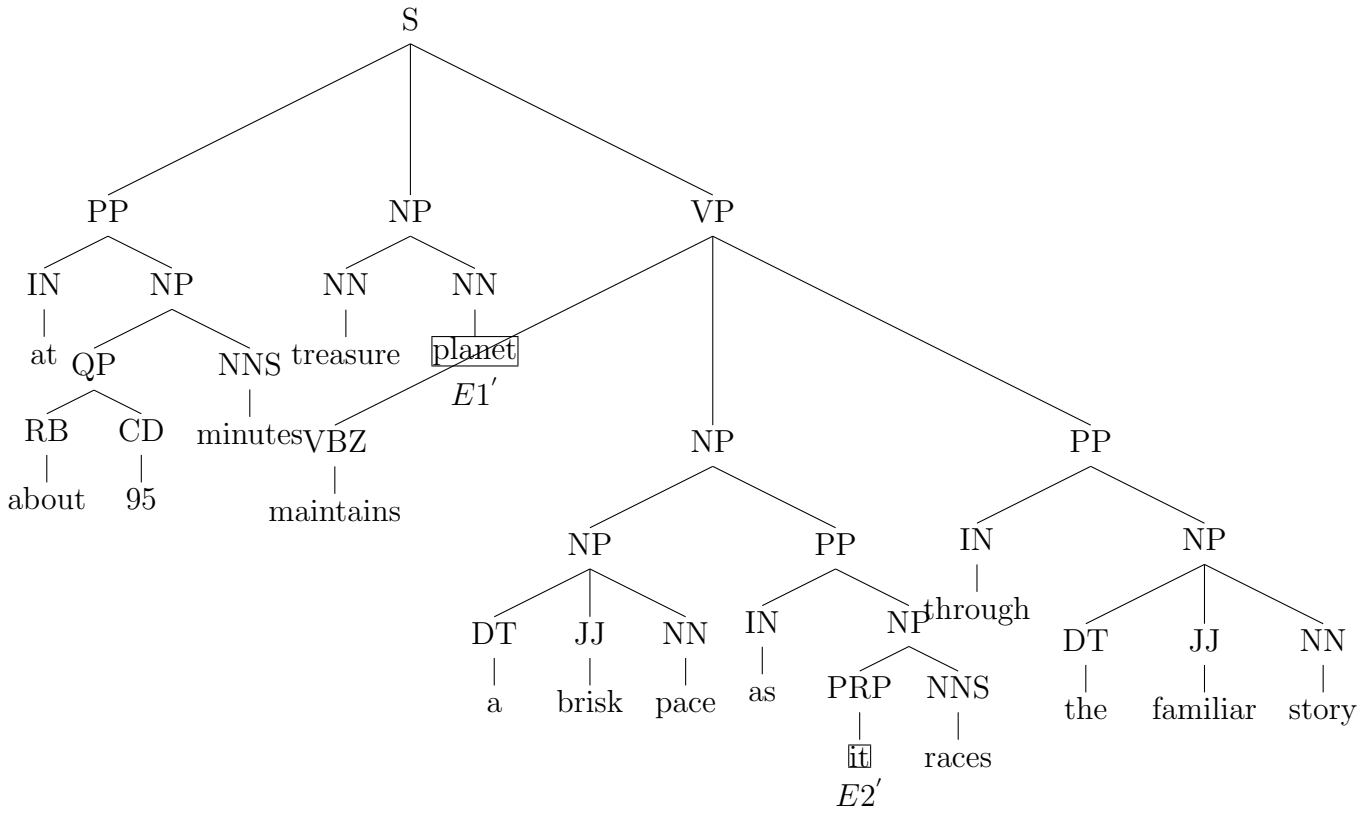

Figure 3.10: *MCT for sentiment word "brisk" using adjective based pruning strategy*

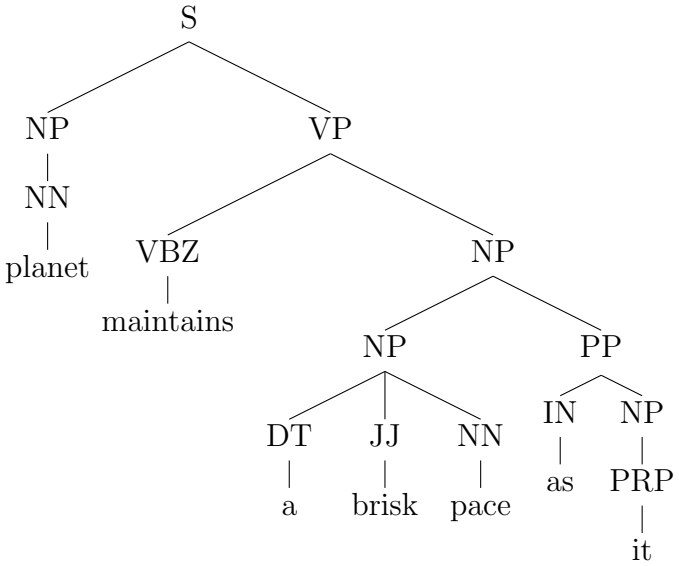

Figure 3.11: *PT for sentiment word "brisk" using adjective based pruning strategy*

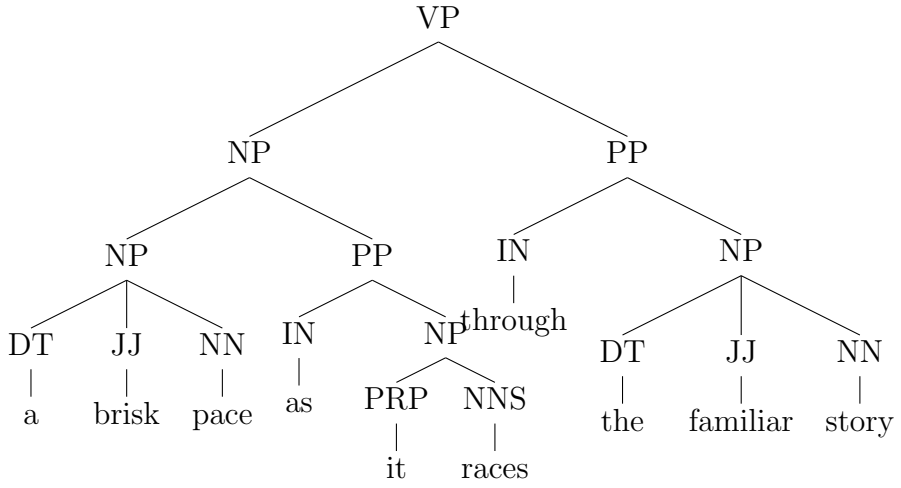

Figure 3.12: *MCT for sentiment word "familiar" using adjective based pruning strategy.*

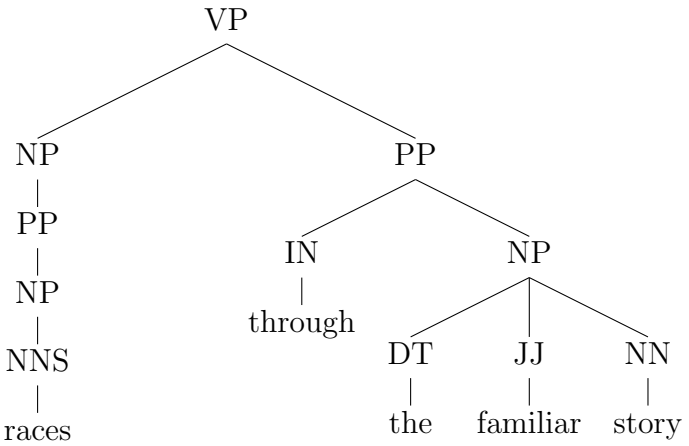

Figure 3.13: *PT for sentiment word "familiar" using adjective based pruning strategy.*

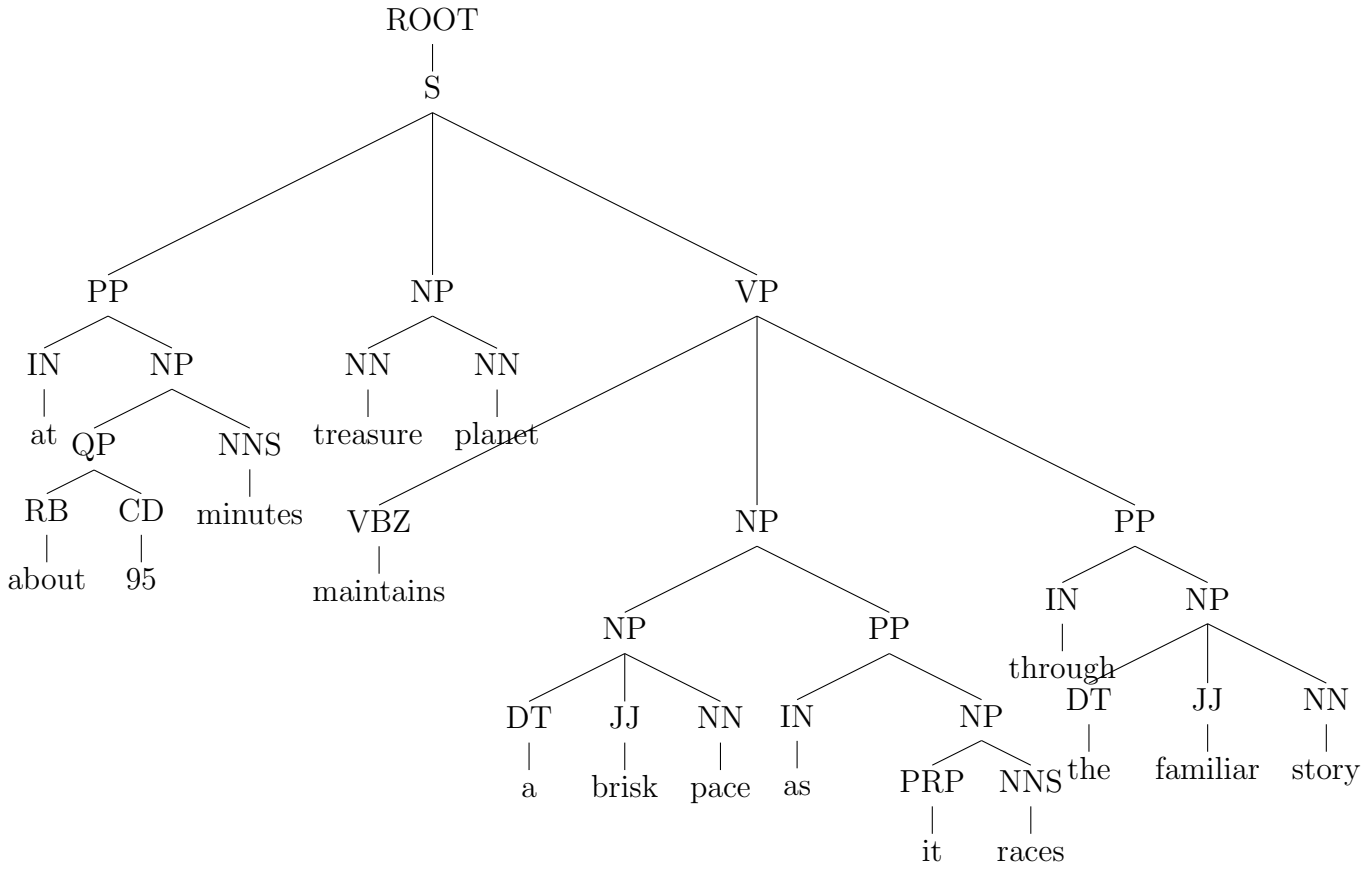

Figure 3.14: *Combined MCT using adjective based pruning strategy*

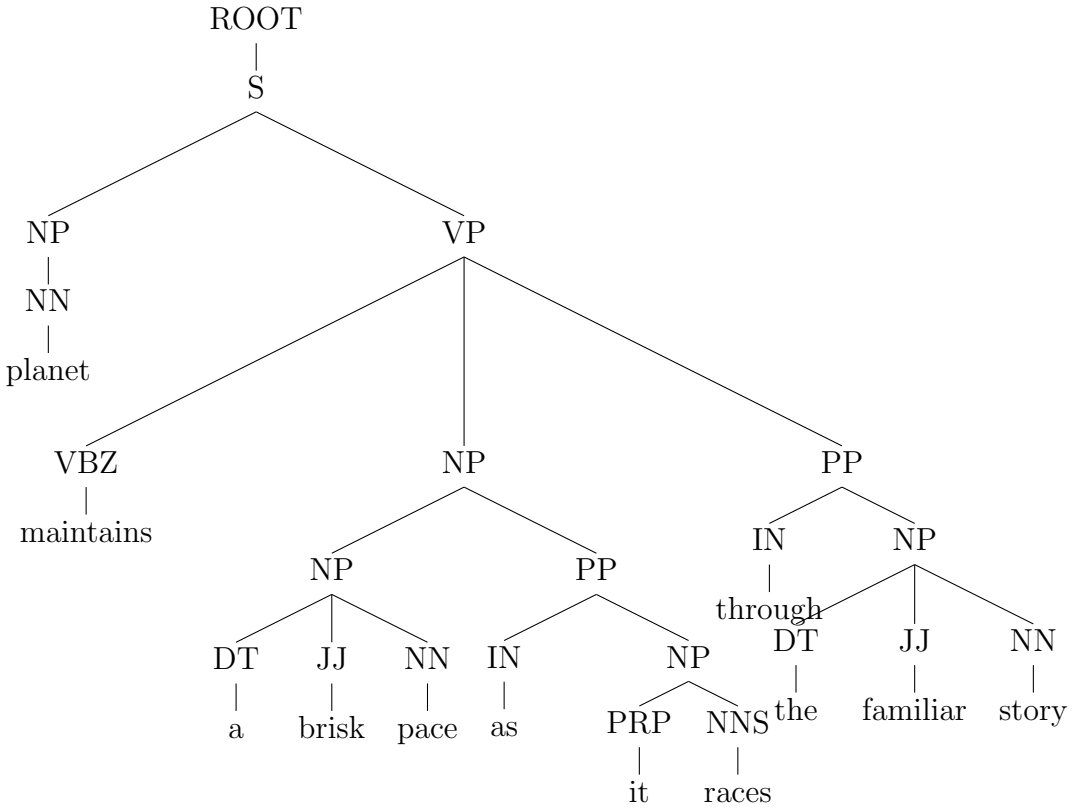

Figure 3.15: *Combined PT using adjective based pruning strategy*

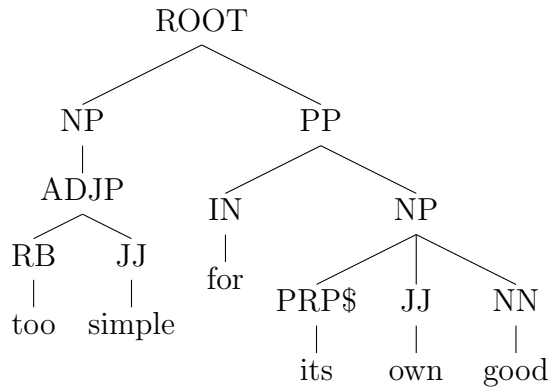

Figure 3.16: *Syntax tree generated using the Stanford parser*

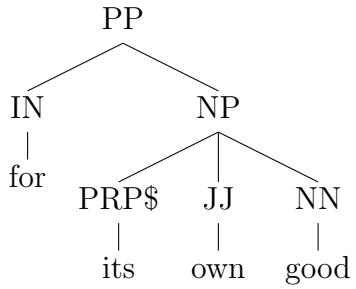

Figure 3.17: *PT subtree generated using sentiment based pruning strategy*

Figure 3.18.

3.4.1 Grams based on Complete Syntax Trees

The following are various types of subtrees (grams) obtained from the complete syntax tree and used as features in our problem.

- **All grams with leaf nodes:** This type of feature representation has all the possible parent-child subtrees as features, as shown in Figure 3.19.
- **Unigrams with leaf nodes:** This type of feature representation contains unigram subtrees as features, as shown in Figure 3.20. Unigram consists of one item for any given sequence of words/input. An n-gram consists of n items for any given input. Let's assume that we have the following input: "too simple for its own good." The unigrams at word level for the above sentence are: too, simple, for, its, own, good.

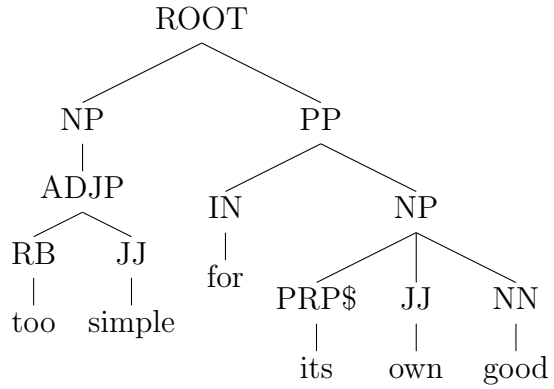

Figure 3.18: *PT subtree generated using adjective based pruning strategy*

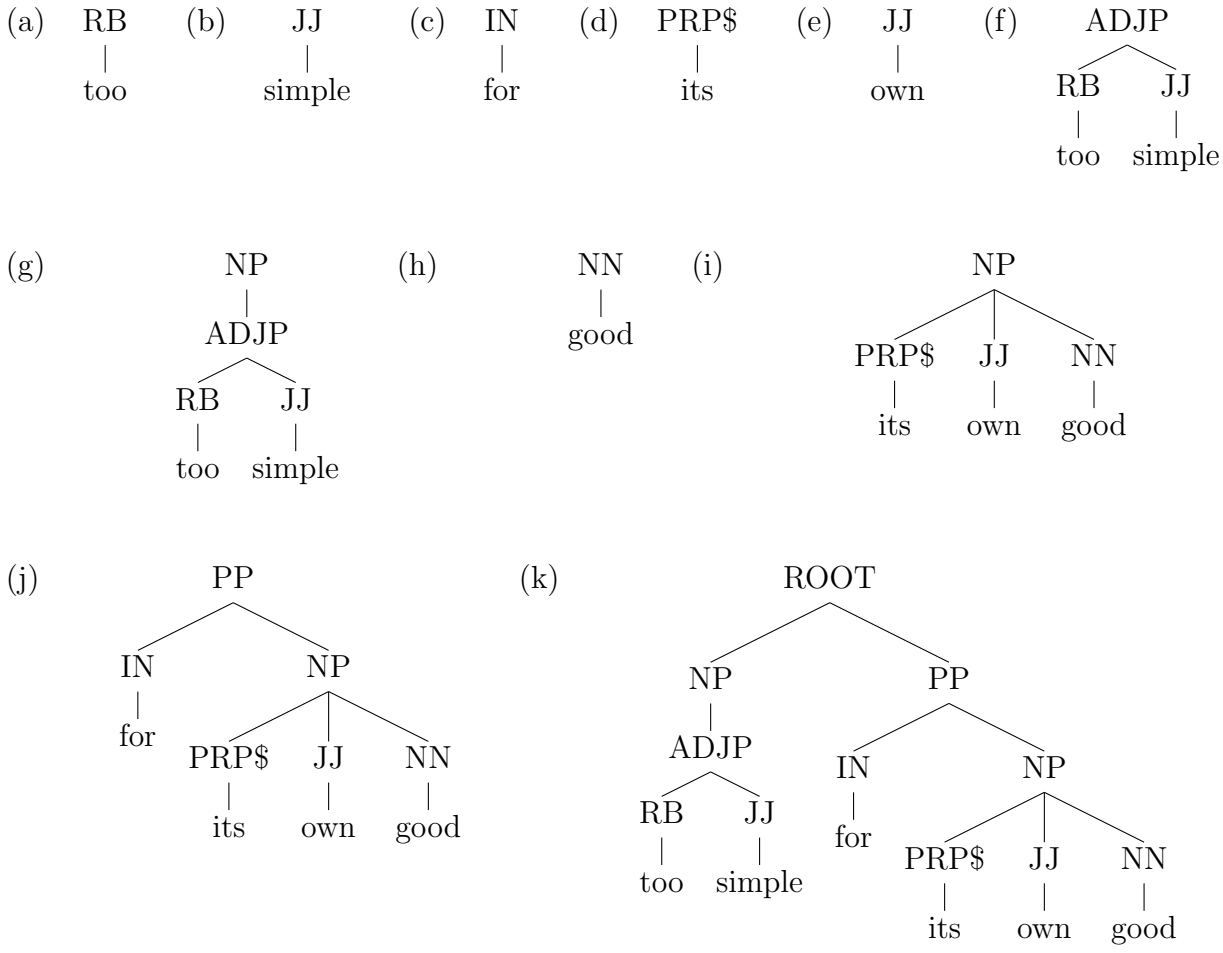

Figure 3.19: *All grams with leaf nodes*

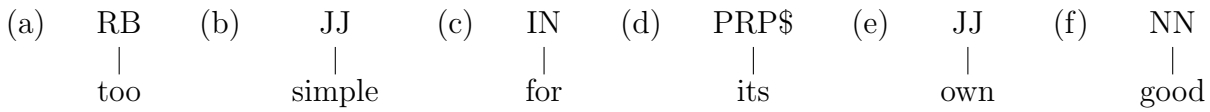

Figure 3.20: *Unigrams with leaf nodes*

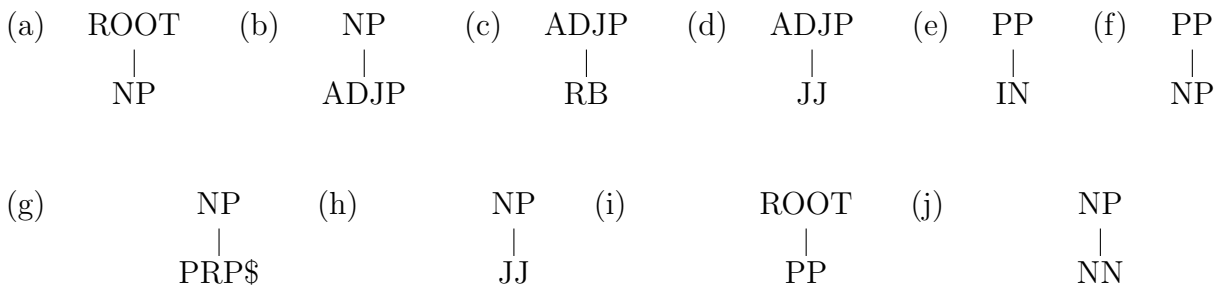

Figure 3.21: *Unigrams without leaf nodes*

The bigrams at word level for the above sentence are: too simple, simple for, for its, its own, own good. The n-gram at the word level is: too simple for its own good. Here, unigram subtrees are just a pair composed of a parent and their child node, where the child node is a leaf node.

- **Unigrams without leaf nodes:** This type of feature representation contains all possible unigram subtrees as features except the unigrams with leaf nodes, as shown in Figure 3.21. Here, unigram subtrees are just a pair composed of a parent and their child node, where the child node is not a leaf node.

- **All unigrams:** All possible unigrams present in the syntax tree are taken as features, as shown in Figure 3.22. The combination of unigrams with leaf nodes and unigrams without leaf nodes gives all possible unigrams.

3.4.2 Grams based on Pruned Syntax Subtrees

As we have seen earlier, the PT tree based on sentiment word based pruning strategy is shown in Figure 3.17. Next, we retrieve the different types of grams exactly the same way as described in Section 3.4.1 except that we have to use PT tree instead of complete (or full) tree. The PT trees based on adjective word based pruning strategy is shown in Figure 3.18.

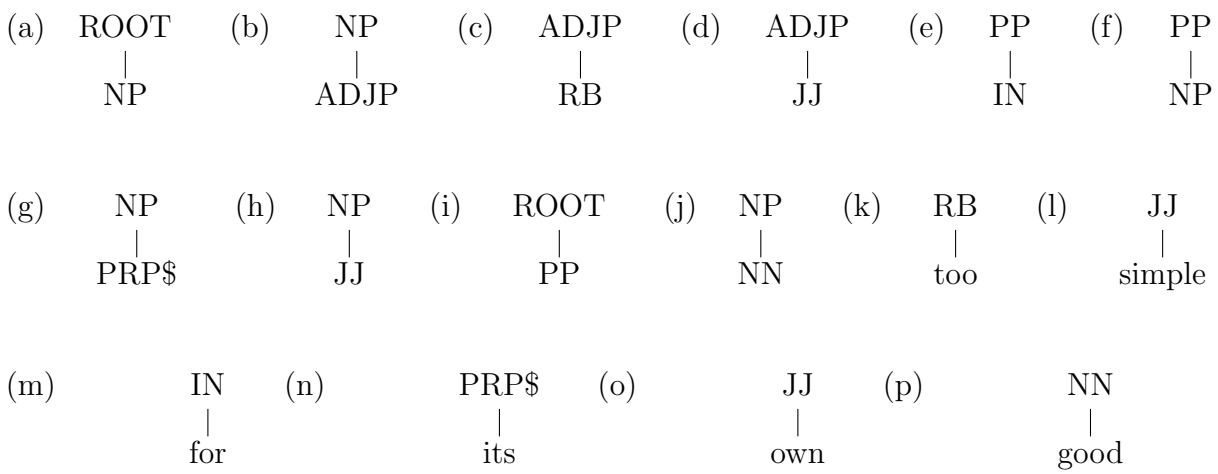

Figure 3.22: *All unigrams*

We retrieve the different types of grams exactly the same way as described in Section 3.4.1 except that we use the PT tree instead of the complete tree. We can use the complete syntax trees but we are identifying MCT and PT trees because we would like to reduce the noise, a task that is not important for sentiment classification problems.

3.5 Feature Construction for Domain Adaptation

We can select any of these types of grams as features based on their predictive power evaluated using a supervised algorithm in a single domain. Now, the main challenge for performing domain adaptation is to select the features to build a bridge between the source domain and target domain. These features are known as domain independent features. Let us look at the following definitions before discussing about the approach that we used to select the features for building a bridge between the two domains. In our problem we have two domains, source domain and target domain. Each domain has two different kinds of features: domain specific feature and non-domain specific or domain independent feature. These two types of features may be either sentiment or non-sentiment words.

3.5.1 Domain Specific Features

Domain specific features are those features with specific meaning in either source or target domain. For example: words like thrilling, powerfully acted express sentiment in the movie domain and words such as exceptional control, sleek and distinctive, space-efficient express sentiment in the kitchen appliances domain.

3.5.2 Domain Independent Features

Domain independent features are those features, which have the same meaning in both source as well as target domain. For example: words like good, excellent, bad, worse, etc. are domain independent features.

Why are we interested in domain independent features?

As mentioned earlier domain independent features have the same meaning in both domains 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. Specifically, we 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.

How do we extract domain independent features?

We used a 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 following formula proposed by 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 denotes the total number of instances in the source domain and target domain, respectively. We have used a constant α to avoid any overflow. In our work, α value is set as 0.0001. To avoid the divide by zero error when both the source domain and the target domain probabilities are same, a constant factor β is introduced. After introducing a constant β, the above formula is modified as follows:

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

In our work, β value is set to 0.0001.

3.6 Approaches Used

The following are the various types of machine learning algorithms that are most widely used for classification tasks:

3.6.1 Supervised Learning Algorithms

Supervised algorithms require labeled data, and are very useful when we have a sufficient amount of labeled data to learn a good classifier. In a supervised framework, for a binary problem, we provide a set of training examples, where each example can belong to either a positive class or a negative class. The set of training examples are used to train a model. This

model will be used to predict new test instances as positives or negatives. The performance is evaluated by comparing the predicted values to the actual values on a hold out dataset. Examples of supervised learning algorithms include: Support Vector Machine (SVM), Naïve Bayes Multinomial (NBM).

Support Vector Machines (SVM)

SVM algorithm is a supervised machine learning algorithm that works very well, especially for binary classification problems. SVM represents each input example as a point in a high dimensional space. If the data is (almost) linearly separable, SVM uses these example points to constructs a hyperplane that separates the positive points from negative points. We can construct many hyperplanes for a given set of example points. The best hyperplane is the one which has the largest separation gap between the positive examples and the negative examples, and this is the hyperplane that SVM finds. New examples are classified to either one of the classes based on which side of the separating hyperplane they fall in, as described in [Cortes and Vapnik, 1995]. When data is not linearly separable in the original space, kernels are used to map the data to a higher dimensional space where data becomes linearly separable. In our work, we used SVM with a tree kernel and a linear kernel in the supervised scenario.

The kernel-based algorithms automatically select the substructures that better describe the subtrees. If the given set of instances are represented in a high dimensional space Z, then the kernel function K is defined as $R : Z \times Z \rightarrow [0, \infty]$, a function which maps a pair of instances $x, y \in X$ to their similarity score $K(p, q)$. In the following few lines we discuss linear and tree kernels:

1. A linear kernel is the default kernel used by SVM and is very useful when we have many attributes. It is defined as the inner product of the two variables as described in [Cortes and Vapnik, 1995]: $R(x, y) = \langle x, y \rangle = \sum_i x_i \cdot y_i$

2. A tree kernel [Souza, 2010] is used in order to find the similarity between two syntax trees by calculating the dot product of feature vectors in high (or even infinite)

dimensional feature spaces.

In our work, we use SVM and compare the performance of tree kernels with linear kernels for sentence level sentiment classification in a single domain. Our initial aim is to capture structured information in terms of sub-structures, which acts as an alternative to flat features. To extract the syntactic structured features embedded in a complete parse tree, we use pruning strategies like sentiment word based and adjective word based strategies as described in Section 3.3. In our work we use a tool called SVM-light-tk [Moschitti, 2002] which implements tree kernels.

SVM-light-tk has the tree kernel implementation inside SVM-light developed by Joachims [2002]. SVM^{light} is an implementation of Support Vector Machines in C language. It is used for solving binary classification problems. SVM^{light} consists of two modules: a learning module (SVM-learn) and a classification module (SVM-classify). SVM-learn builds a model from the training data. SVM-classify reads this model file and makes predictions on test data instances.

Naïve Bayes Multinomial (NBM)

Naïve Bayes Multinomial is a supervised learning algorithm and it is widely used for document level classification tasks. It builds the model by using a set of labeled training examples and uses this model to classify the new unlabeled instances. Naïve Bayes Classifier is a probabilistic classifier and it is based on a strong independence assumption (words in a document are independent given the class label of the document). Furthermore, positions in a sentence are also independent. The NBM algorithm is found to be very useful for text classification, when simply words are used as features. For example, a movie can be classified as a comedy movie if it contains words such as funny, joyful, or humor character names such as Ben Stiller etc. Even though words in a sentence are not dependent, naïve Bayes classifier considers all these words as independent features. In our experiments, we assume that the probability of a gram occurring in a document is totally independent of the gram's position and its context in a sentence, given the class label of a sentence. In what

follows, we provide more details of the algorithm.

Let us assume that we are given a document D to classify as either positive or negative class c_k, where $c_k \in \{+1, -1\}$. Using the independence assumption, a document can be seen as a bag of words $(w_1, w_2, \cdots, w_n) \in D$ and the word in a document is independent of its corresponding position and context in a document. Naïve Bayes classifier is based on the Bayes theorem and learning the classifier can be reduced to estimating class prior and the data likelihood. Then, the probability of a class given a document $P(c_k|D)$ (posterior probability) is proportional to multiplying the prior $P(c_k)$ and the probabilities $\prod_{v=1}^{n} P(w_v|D)$. Finally, we check the probability distributions of $P(c_k|D)$ for all values of k and then classify the given document in the class which has the highest probability distribution value. The probability of a word given class can be estimated using the following formula as described in [Tan et al., 2009]:

$$P(w_v|c_k) = \frac{N_v * P(c_k|D) + 1}{\sum_{v=1}^{|V|} N_v * P(c_k|D) + |V|}$$

where w_v is a word from a given vocabulary (V) considered as features, c_k is the class label where $c_k \in \{+1, -1\}$, N_v is the number of times word has occurred in a given document D and $|V|$ is the size of the vocabulary (that contains all the words $(w_1, w_2, \cdots, w_n)$).

We can easily calculate the prior probability in a given document by counting the number of documents with different class labels. The prior probability is calculated using the following formula:

$$P(c_k) = \frac{\text{Number of examples in } c_k}{\text{Total number of examples}}$$

Next, $P(c_k|D)$ is approximated by multiplying both the prior and the posterior probabilities as follows:

$$P(c_k|D) \approx P(c_k) * \prod_{v=1}^{n} P(w_v|c_k)^{N_v}$$

In general, NBM works for a collection of documents. In our work, we used sentences instead of whole documents.

3.6.2 Domain Adaptation Algorithms

Our goal is to reduce the gap between the source and target domains by learning a classifier from source domain and target domain instances to predict the labels for the new target domain unlabeled instances. We use domain adaptation algorithms when we have labeled data in the source domain and little or no labeled data along with unlabeled data in the target domain. We consider two domain adaptation scenarios, as described in what follows:

1. **Case 1**: *Data available is source domain labeled data and only unlabeled data from target domain.* In this case, we build a classifier using the source domain labeled data represented with generalized features only. We use the source domain classifier in order to predict the corresponding labels for the target domain unlabeled instances. Next, we build the target domain classifier by using the predicted labels for target domain represented with the whole vocabulary from the target domain as its features. Finally, we use both the source domain classifier and target domain classifier to predict new labels for the unlabeled instances from the target domain and we repeat this process until we meet a convergence point.

2. **Case 2**: *Data available is source domain labeled data, and a small amount of labeled data along with unlabeled data from target domain.* In this case, first we build a combined classifier using the source domain labeled data and target domain labeled data. Next, we use this classifier in order to predict labels for the unlabeled target domain instances. After predicting the labels for the target domain unlabeled instances, we build another combined classifier using the source domain labeled data, target domain labeled data and the predicted labels for the unlabeled target domain data to predict the labels for the target domain unlabeled instances. This process is repeated until we meet a convergence point.

The Adapted Naïve Bayes algorithm used to address these two cases is discussed below. The original algorithm works for the first case, but we also adapted it for the second case.

Adapted Naïve Bayes (ANB)

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 Expectation Maximization (EM) on top of naïve Bayes classifier, to predict the target domain unlabeled instances. One important part of the ANB algorithm is that it simultaneously allows us to reduce the weight given to the source domain instances, while increasing the weight given to the target domain instances at each iteration, by using a constant lambda (λ). This, in turn, allows us to predict the labels for the target domain instances.The EM algorithm is used to find the maximum likelihood. It 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) and the model parameters given the observed or known data. In the M-step, we try to maximize the likelihood function by assuming that the missing data is known. The two steps are repeated until we reach a convergence point.

The EM approach used in ANB and described in [Tan et al., 2009] is different from the traditional EM approach because we want to find the maximum likelihood only for the target domain instances but not for the source domain. ANB algorithm achieves this by increasing the weight for the target-domain data, while decreasing the weight for the source-domain data at each iteration. Also, remember that we do not use all the features available in the source domain. We use only a small amount of features also known as generalized features from the source domain because our goal is to classify the target domain instances, but not the source domain. So, in each iteration we use only generalized features for the source domain, whereas we use the whole vocabulary for the target domain. This helps us to improve the prediction ability of the classifier to classify the target domain instances because the domain specific features from the source domain are not very useful for predicting the target domain instances.

In the following lines, we give the detailed formulas under domain adaptation setting using the ANB classifier on top of EM as described by Tan et al. [2009]:

E-step:

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

M-step:

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

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

1. Case 1: During the first iteration $D_t \in D_{t\_lab}$ in the above M-step. From the second iteration onwards $D_t \in D_{t\_lab}, D_{t\_unlab}$ until we reach a convergence point.

2. Case 2: During the first iteration $D_t = \phi$ in the above M-step. From the second iteration onwards $D_t \in D_{t\_unlab}$ until we reach a convergence point.

where $N_{v,k}^s$ and $N_{v,k}^t$ denote the number of appearances of feature f_v in source domain and target domain for its corresponding class c_k. These are obtained as follows:

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

where λ is a parameter for controlling the weights for the source domain versus target domain. The value of λ changes with the number of iterations (τ), which is expressed as: $\lambda = \min\left(\delta * \tau, 1\right)$ and $\tau \in \{1,2,3,...\}$ until we reach the convergence point. Here, δ is a constant and in our work we used $\delta = 0.2$. η_v^s is a constant and is given as:

$$\eta_v^s = \begin{cases} 0 & \text{if } f_v \notin V_{FCE} \\ 1 & \text{if } f_v \in V_{FCE} \end{cases}$$

The time complexity for ANB algorithm is O(nm), where n is the number of examples and m is the number of attributes. However, the complexity of the EM step is O(nmk), where k is the number of iterations.

Chapter 4

Experimental Setup

In this chapter, we explain the research questions that we have addressed in our work, the dataset used and the experiments designed to evaluate our approach. We have conducted experiments with various classifiers and with variable amount of data to investigate the performance of the classifiers for domain classification within a single domain and across domains. Specifically, this chapter is organized as follows: In Section 4.1, we describe various research questions that we have addressed. In Section 4.2 we list the set of experiments that have performed performed. In Section 4.3, we describe the dataset used.

4.1 Research Questions

The following are the research questions that we have addressed in this work:

- Are the grams extracted from syntax trees, when used as features in a domain specific classifier, comparable in terms of prediction and accuracy to the structured syntax subtrees? Overall, how useful are these features for the sentence-level sentiment classification problem?

  According to Zhang et al. [2010], structured features give very good results in terms of accuracy for sentence-level sentiment classification. However, for our domain adaptation algorithm, ANB, we need to provide features in terms of gram counts. That is possible, when considering the gram based features described in Section 3.4. However,

we would like to know how good these grams are with respect to the sentiment classification problem we are addressing. To evaluate the prediction power of these grams, we compare the results of SVM classifiers using grams extracted from trees, in a single domain scenario, with the results obtain based on SVM with structured syntax tree features.

- What is the effect of using various gram types described in Section 3.4 (i.e., all grams with leaf nodes, unigrams with leaf nodes, unigrams without leaf nodes, all unigrams) for training the domain adaptation classifiers? Is it better to use all the features or a reduced set of features? Also, how many features do we need to use as generalized features?

  According to Pan et al. [2010], we need to identify a predefined set of words, known as domain independent features, before performing cross-domain sentiment classification. We extract domain independent gram features (extracted from syntax trees generated from a Stanford parser) based on an entropy calculation method. We evaluate the performance of sentiment classification across domains by considering the top 50 FCE target grams or top 100 FCE target grams as generalized features. We also experiment with different size gram vocabularies. First, we use all the grams present in a target domain. We have also experiment with a reduced vocabulary obtained by removing the grams that occur 1 time, 2 times or 3 times.

- How does the ANB approach perform when using some target domain labeled data versus not using any target domain labeled data? Are the results better when using some target labeled data?

  We would like to compare the performance of domain adaptation classifiers learned using source domain labeled data, target domain labeled data and target domain unlabeled data, versus the performance of classifiers learned using only source domain labeled data and target domain unlabeled data.

- How does the ANB approach perform when compared to supervised NBM classifiers?

  We build a supervised naïve Bayes classifier with target domain labeled as training data and tested it on target domain unlabeled. Next, we build another supervised naïve Bayes classifier, where all the target domain data (labeled and unlabeled) is used as labeled data. This can be seen as an upper bound for our approach. We have also built another supervised naïve Bayes classifier, with source domain as training data and target domain unlabeled as test data. This is considered as the lower bound for our approach. All the above supervised approaches are tested on the same test dataset that is used for the ANB approach. Therefore, we compare the results obtained from ANB with the results obtained using the supervised naïve Bayes classifiers.

4.2 Experiments

There are two types of experiments that we have performed in our work. First, we learn a domain specific classifier in a given domain and then test it on unlabeled instances from the same domain. Second, we learn a domain adaptation classifier under the assumption that there is little or no labeled data in a target domain, and labeled data from a source domain might help. Thus, the domain adaptation classifier is trained on a combination of labeled source and labeled/unlabeled target data, and is used to predict labels on target domain unlabeled data. The following description gives amore details on the what type of experiments we have performed for single and cross-domain classification tasks.

4.2.1 Domain Specific Classifiers

Experiment 1: The purpose of this experiment is to evaluate the performance of the SVM algorithm on sentiment classification by using tree kernels along with structured features obtained by pruning strategies. The models are built using the structured features as described in Section 3.3. These experiments are conducted using tree kernels along with MCT and PT trees. To have a baseline, we have also conducted experiments using the unigrams

as features. The unigrams are obtained by eliminating all the stop-words. SVM-light-TK is used for model building and all the results are obtained via 3-fold cross validation. Experiment 1 compares the performance of the results between unigrams and tree kernels methods. Precision, Recall and F1 measure values averaged over 3 folds are used to evaluate the results. The dataset that we have used for this experiment is described in Section 4.3 and the experiments are performed only in the movie domain with 10662 (5331 positive and 5331 negative) instances.

Experiment 2: The purpose of this experiment is to evaluate which types of gram representations work the best in a domain specific task, with the goal of using such grams for cross-domain classifiertion as well.

a) Here we used SVM classifier to classify the sentences in a single domain with different kinds of grams as our features (see Section 3.4) and then compared the results with the results that we have obtained using SVM with structured syntax subtrees as features. These experiments were conducted on the movie domain with 1800 (900 positive and 900 negative) instances. The results allow us to understand which types of grams are useful for the classification tasks considered in this work.

b) We have also used a naïve Bayes classifier to classify the sentences in a single domain with different kinds of grams from PT trees (obtained using sentiment based pruning strategy) as features. Next, we compared the results with the results obtained using SVM with structured trees or different kinds of grams from complete syntax tree as features. We have conducted these experiments on the movie domain with 1800 instances. We use naïve Bayes because we use an adapted naïve Bayes classifier in cross-domain classification and we would like to compare our results using supervised naïve Bayes classifier. As said earlier, these results will help us in deciding which types of grams are useful for our classification tasks.

As we will see in the results chapter, classification results using "all grams with leaf nodes" and "unigrams with leaf nodes" as features along with SVM are very similar to the

results obtained using "structured trees" as features using SVM classifier. This suggests, that using "all grams with leaf nodes" and "unigrams with leaf nodes" as features might give better results than using "unigrams without leaf nodes" and "all unigrams" as features for sentiment classification across domains.

4.2.2 Domain Adaptation Classifiers

In the cross-domain scenario, we perform two sets of experiments using various types of grams as features.

1) In the first set of experiments, we use different types of grams from the complete syntax tree as features (as described in Section 3.4) for our domain adaptation classifiers.

2) In the second set of experiments, we first prune the complete syntax trees to obtain PT trees using the sentiment word based pruning strategy. Then, we select various types of gram representations from the PT trees as features in our domain adaptation classifier.

Experiment 3: The purpose of this experiment is to study the necessity of using all the grams as opposed to reducing the number of grams based on frequency. We also identify the number of generalized features that we need to consider in the source domain. For this, we select the grams that result in the highest precision and recall values in Experiment 2. Next, train models using different types of gram features on various customer review datasets (movies, DVDs, kitchen appliances). Before training the classifiers, we perform a filtering stage in order to remove some of the uninformative grams from source and target domains. For this we perform various experiments using all the grams or grams that occur more than 1 time or 2 times or 3 times as vocabulary for target domain, along with top 50 or 100 FCE grams as generalized features. Before going through the experiments in this category, let us discuss the steps involved in these experiments.

Source and Target Domains: Our customer review dataset includes three different domains: movie domain (M), DVDs domain (D) and kitchen appliances (K) domain. So,

the possible source and target combinations are $D \rightarrow M, M \rightarrow D, M \rightarrow K, K \rightarrow M, D \rightarrow K, K \rightarrow D$. The left side of the arrow represents the source domain and the right side of the arrow represents the target domain.

Steps Performed in Experiment 3: For all possible combinations, we first extract the syntax trees for each sentence in both source as well as target domains. Next, we identify the corresponding counts for each and every gram in source and target domains separately. As explained above, we conduct various experiments by considering all the grams or by eliminating those grams that occur only 1 time, 2 times and 3 times in either domain. Next, we calculate the entropy values for all the grams present in the target domain using the frequently co-occurring entropy (FCE) method described in [Tan et al., 2009]. After finding the FCE values using the number of occurrences of grams in both the source and target domains, we will consider the top 50 or the top 100 grams (with the highest FCE values) as generalized features. We use all the grams present in the target domain as our vocabulary for the target domain. We have performed 3 fold cross-validation on target domain data. Preliminary results suggest that it is better to remove all the grams that are occurring 1 time and also to consider using top 100 FCE features as generalized features.

Classifiers Compared in Experiment 3: The following are the cases that we consider for each experiment before coming to the above conclusion (i.e., remove all the grams that occur 1 time and consider 100 features as generalized features):

- Case 1: First, we use ANB classifier to perform cross-domain sentiment classification using source domain labeled data and target domain unlabeled data. As explained earlier, we perform 3 fold cross-validation on target domain. We consider 2 folds of the target domain (unlabeled data) along with source domain (labeled data) as our training data and use the remaining one fold of the target domain unlabeled data as test data. This is considered as the lower bound for our experiments.

- Case 2: Next, we use ANB classifier to perform cross-domain sentiment classification using source domain labeled data, together with target domain labeled and unlabeled

data. In this experiment, when we consider target domain labeled data, we split the target domain (movie domain with 400 instances) or (DVDs domain with 800 instances) or (DVDs domain with 400 instances) or (kitchen domain with 400 instances) as 100 labeled target domain instances and the remaining as unlabeled target domain instances. We consider 2 folds of the target domain (labeled and unlabeled data) along with source domain (labeled data) as our training data and use the remaining one fold of the target domain unlabeled data as our test data. So, the number of test instances are one-third of the total number of unlabeled target domain instances.

- Case 3: In addition to the above two experiments, we also perform two supervised sentiment classifications. In this case, we perform a supervised (naïve Bayes multinomial) sentiment classification with target domain labeled instances as training data and target domain unlabeled instances as test data. These experiments are also performed using 3 fold cross-validation on target domain.

- Case 4: We perform another supervised classification (naïve Bayes multinomial), with target domain labeled plus target domain unlabeled instances (assumed labeled as well) as training data and target domain labeled instances as test data. We consider this as the upper bound for our experiments. These experiments are performed using 3 fold cross-validation on target domain as well.

Unigrams with Leaf Nodes as Features: The following experiments are performed with source as movie and target as DVDs. We have also used source as DVDs and target as movie. There is another special case that we have considered to have with source as movie (400 instances) and target as DVDs (400 instances). The goal of these experiments is to learn what sets of grams are more predictive. Based on the experimental results for each experiment for the above 4 cases, we decide whether to include the set of all the grams or a reduced set of grams (based on frequency).

1. In this experiment we consider all the grams as features. We consider the top 50 FCE features as generalized features for representing the source domain and all the grams present in the target domain as our vocabulary for the target domain. As a special case, we performed another experiment as described above, except that we used the top 100 FCE features as generalized features for representing the source domain.

2. In this experiment we consider grams that occur more than 1 time. Here we consider the top 50 FCE features as generalized features for representing the source domain and all the grams present in the target domain as our vocabulary for the target domain. We performed another experiment as described above, except that we used the top 100 FCE features as generalized features for representing the source domain.

3. In this experiment we consider grams that occur more than 2 time. Here we consider the top 50 FCE features as generalized features for representing the source domain and all the grams present in the target domain as our vocabulary for the target domain. We performed another experiment as described above, except that we used the top 100 FCE features as generalized features for representing the source domain.

From the above experimental results, we observed that it is better to remove all the grams that are occurring 1 time and also to consider using top 100 FCE grams as generalized features. So, for all the remaining experiments with various source and target combinations along with different gram representations, we consider grams>1 in source and target domains with the top 100 FCE grams as generalized features.

Experiment 4: The purpose of this experiment is to to compare the performance of sentiment classification using ANB (domain adaption classifier) and supervised naïve Bayes algorithm (domain specific classifier) across all possible combinations of source and target domains. Here, we compare the results of ANB and naïve Bayes by performing 3 fold cross-validation on target domain. We also compare the performance of ANB classifier for cross-domain sentiment classification by using a little amount of target domain labeled

data versus ANB classifier without any target domain labeled data. In addition to the four different cases mentioned in experiment 3, we perform another supervised classification (naïve Bayes multinomial), with training as source domain labeled data and target domain unlabeled instances as test data represented as Case 5 in our experiments. This is considered as the lower bound for our experiments and we perform a 3 fold cross-validation on target domain.

Experiment 5: The purpose of this experiment is to evaluate the performance of domain adaptation for cross-domain sentiment classification by using ANB with various types of grams as features from PT (structured syntax subtrees obtained by using sentiment word based pruning strategy) trees. So, we perform all combinations of source and target domain as specified in Experiment 3, with various types of grams as features from structured syntax tree (PT tree) as our features.

4.3 Data Description

This section gives an overview of the data that we used in our experiments. We used two different datasets. One dataset is the movie review dataset obtained from Pang and Lee [2005], which contains 5331 positive sentences and 5331 negative sentences. We have manually constructing the second dataset by extracting reviews from Amazon and BestBuy. In order to extract the data from BestBuy we have used the BestBuy API package at `https://bbyopen.com/developer`. After creating an account, they provide us a unique *apikey* so that we can use the key to retrieve the necessary information. But we cannot directly get the reviews from BestBuy. First, we use a command to retrieve the sku numbers for the reviews with rating>3 or reviews with rating<3 for a given product. Next, we use another command to retrieve the reviews of the products using the sku numbers obtained from above. We select reviews with rating>3 because generally reviews with rating 4 or 5 are considered as positive reviews. And we select reviews with rating<3 because generally reviews with rating 1 or 2 are considered as negative reviews. We have manually examined each sentence

to make sure that we include a sentence with a positive sentiment in positive reviews class and a sentence with negative sentiment in a negative reviews class. The dataset contains customer reviews of products such as movies, DVDs and kitchen appliances.

The above extraction procedure resulted in 200 positive sentences and 21 negative sentences for the movie domain from BestBuy. For the DVDs domain, we extracted 400 positive and 400 negative sentences and for the kitchen appliances domain, we extracted 192 positive sentences and 79 negative sentences from BestBuy. But we want our dataset to have an equal amount positive and negative sentences (as we didn't aim to evaluate the effect of unbalanced dataset on the results). So, we also crawled through Amazon customer reviews and extracted 179 negative sentences for the movie domain. We have also extracted 8 positive sentences and 121 negative sentences for the kitchen appliances domain from Amazon. Finally, we have 400 (200 positive, 200 negative) instances in the movie domain (M), 800 (400 positive, 400 negative) instances in the DVDs domain (D) and 400 (200 positive, 200 negative) instances in the kitchen appliances (K) domain. These numbers are summarized in Table 4.1. As we have two datasets for the DVD domain, one containing 400 instances and the second one containing 800 instances, we will denote these datasets by D (400) and D' (800), respectively, and will use these notation from now on. As above, the movie dataset is denoted by M and the kitchen appliances dataset is denoted by K.

Table 4.1: *Customer review dataset*

Customer review product sentences			
Product	Total No. of sentences	No. of pos. sentences	No. of neg. sentences
movie	400	200	200
DVDs	800	400	400
kitchen appliances	400	200	200

In our work, we use the movie review dataset from Pang and Lee [2005] (consisting of 5331 positive sentences and 5331 negative) for building a domain specific classifier and the

customer review dataset for building the domain adaptation classifier. For Experiments 1 and 2, we used 10662 (5331 positive and 5331 negative) sentences from the movie review dataset by Pang and Lee [2005] as the experimental dataset. We use SVM-light-tk tool for this experiment. To identify words, the sentiment words dictionary, available at http://www.cs.uic.edu/ liub/FBS/sentiment-analysis, html was used. The sentiment word list was first developed and used by Hu and Liu [2004].The sentiment words dictionary consists of 6789 sentiment-words. For the adjective words, we used the online list of adjective words which contains about 1141 at http://www.enchantedlearning.com/wordlist/adjectives.shtml. For single domain, we also performed experiments using unigrams as features. Unigrams are nothing but one grams and they are singular words present in sentences from the dataset. Based on unigrams, we look at each sentence to identify the presence of those words. For unigrams, we consider the list of words present in the experimental dataset (i.e, movie review dataset by Pang and Lee [2005]) as the vocabulary, after eliminating all the stop-words.

For Experiments 3, 4 and 5, we have customer reviews as our dataset. For Experiments 3 and 4, a detailed description of the total number of grams in the customer review dataset obtained from a complete syntax tree for each sentence are shown in Table 4.2, Table 4.3, Table 4.4, Table 4.5 and Table 4.6. For Experiment 5, the total number of grams in the customer review dataset obtained from a path tree (PT obtained by applying sentiment based pruning strategy) for each sentence are shown in Table 4.7 and Table 4.8.

Table 4.2: *Number of all grams with leaf nodes as grams in M, D, D' and K, respectively*

Number	M	D	D'	K
unique grams	8037	9391	18147	9726
unique grams>1	985	1267	2202	1279

Table 4.3: *Number of unigrams with leaf nodes as grams in M and D'*

Number	M	D'
unique grams	2150	3942
unique grams>1	698	1447
unique grams>2	418	940
unique grams>3	304	680
unique grams>4	249	565
unique grams>5	201	478

Table 4.4: *Number of unigrams with leaf nodes as grams in M, D and K*

Number	M	D	K
unique grams	2150	2273	2228
unique grams>1	698	861	869

Table 4.5: *Number of unigrams without leaf nodes as grams in M, D, D', K*

Number	M	D	D'	K
unique grams	231	256	313	249
unique grams>1	171	192	246	184

Table 4.6: *Number of all unigrams as grams in M, D, D', K*

Number	M	D	D'	K
unique grams	2374	2523	4230	2474
unique grams>1	864	1047	1682	1053

Table 4.7: *Number of all grams with leaf nodes as PT grams in M, D, K*

Number	M	D	K
unique grams	5148	6275	6044
unique grams>1	633	796	780

Table 4.8: *Number of unigrams with leaf nodes as PT grams in M, D, K*

Number	M	D	K
unique grams	1362	1519	1414
unique grams>1	415	526	511

Chapter 5

Results

5.1 Experiment 1 Results

The average results using SVM-light-TK tool along with tree kernel based approach on the movie review dataset (5331 positive, 5331 negative sentences) are shown in Figure 5.1. The following notations are used in the figure: FT represents full tree; MCT-SPS represents minimum complete tree obtained using sentiment word pruning strategy and MCT-APS represents minimum complete tree obtained using adjective word pruning strategy; PT-SPS represents path tree obtained using sentiment word pruning strategy and PT-APS represents path tree obtained using adjective word pruning strategy.

Table 5.1: *Results for Experiment 1: domain specific classifiers using SVM and trees*

Features	Positive			Negative		
Trees	Precision	Recall	F1	Precision	Recall	F1
FT	0.7038	0.7009	0.7023	0.7022	0.7052	0.6829
MCT-SPS	0.7460	0.6923	0.7180	0.7131	0.7643	0.7377
PT-SPS	0.7258	0.7342	0.7299	0.7153	0.7065	0.7108
MCT-APS	0.6916	0.6883	0.6894	0.6901	0.6919	0.6916
PT-APS	0.6796	0.6677	0.6751	0.6690	0.6841	0.6763
Unigrams	0.6726	0.7123	0.6917	0.6951	0.6541	0.6738

The experimental results show that PT and MCT results are better than the results of FT, possible because FT might loose some useful information. We can even see that PT is better than MCT because PT reduces additional noise from MCT by pruning the nodes from

the left and the right side of a given tree. As a baseline, we conducted experiments using unigrams. The results show that tree kernel based method is very good when compared to the basic kernel method. We can see that the results obtained using unigrams are very less when compared to those obtained using various tree kernels as shown in Figure 5.1. This shows that tree kernels are very good in sentiment classification by capturing the information in the form of syntax trees.

5.2 Experiment 2 Results

The average results for 3 fold cross-validation using SVM with various types of grams as features on movie review dataset (by Pang and Lee [2005]) from FT and PT trees are shown in Table 5.2 and Table 5.3. These grams are extracted from the movie review dataset (10662 sentences). The grams extracted from PT trees are obtained using sentiment word based pruning strategy. In the figure, FT represents full tree, PT represents path tree, ALN represents all grams with leaf nodes, ULN represents unigrams with leaf nodes and UNLN represents unigrams with no (i.e., without) leaf nodes.

Table 5.2: *Results for Experiment 2: domain specific classifiers using SVM and grams from FT*

Features	Positive			Negative		
Grams	Precision	Recall	F1	Precision	Recall	F1
ALN	0.7406	0.7313	0.7359	0.7344	0.7436	0.7389
ULN	0.7350	0.7204	0.7276	0.7256	0.7400	0.7327
UNLN	0.5837	0.6214	0.6019	0.5952	0.5569	0.5754
All unigrams	0.6765	0.6848	0.6805	0.6809	0.6724	0.6765

The average results for 3 fold cross-validation using naïve Bayes multinomial with various types of grams as features on movie review dataset (by Pang and Lee [2005]) from FT and PT trees are shown in Table 5.4 andTable 5.5. These grams are extracted from the movie review dataset (10662 sentences). The grams extracted from PT trees are obtained using sentiment word based pruning strategy. As mentioned above, FT represents full tree, PT

Table 5.3: *Results for Experiment 2: domain specific classifiers using SVM and grams from PT-SPS*

Features	Positive			Negative		
Grams	Precision	Recall	F1	Precision	Recall	F1
ALN	0.7361	0.7481	0.7312	0.6797	0.6742	0.6606
ULN	0.7190	0.7034	0.7111	0.7090	0.7244	0.7167
UNLN	0.5671	0.6476	0.6046	0.5887	0.5049	0.5435
All unigrams	0.6608	0.6719	0.6663	0.6658	0.6545	0.6600

represents path tree, ALN represents all grams with leaf nodes, ULN represents unigrams with leaf nodes and UNLN represents unigrams with no (i.e., without) leaf nodes.

Table 5.4: *Results for Experiment 2: domain specific classifiers using NBM and grams from FT*

Features	Positive				Negative			
Grams	Precision	Recall	F1	ROC	Precision	Recall	F1	ROC
ALN	0.7610	0.7493	0.7550	0.8316	0.7526	0.7643	0.7583	0.8316
ULN	0.8220	0.7953	0.8083	0.8766	0.8013	0.8270	0.8143	0.8766
UNLN	0.5670	0.6163	0.5906	0.6023	0.5793	0.5286	0.5526	0.6023
All unigrams	0.7883	0.7663	0.7770	0.8506	0.7720	0.7933	0.7826	0.8506

Table 5.5: *Results for Experiment 2: domain specific classifiers using NBM and grams from PT-SPS*

Features	Positive				Negative			
grams	Precision	Recall	F1	ROC	Precision	Recall	F1	ROC
ALN	0.7330	0.8940	0.8026	0.9026	0.8606	0.6583	0.7386	0.9026
ULN	0.7846	0.7793	0.7823	0.8543	0.7806	0.7860	0.7833	0.8543
UNLN	0.5626	0.6006	0.5810	0.5940	0.5716	0.5333	0.5516	0.5940
All unigrams	0.8023	0.7753	0.7890	0.8660	0.7823	0.8090	0.7950	0.8660

5.3 Experiment 3 Results

Irrespective of the source and the target domains, we used the frequently co-occurring entropy (FCE) method to calculate the entropy values for both the source and target domain

in order to select the generalized features for the source domain. We have considered the following cases for all possible combinations of source and target domain and performed a three fold cross-validation for each experiment:

1. Case 1: Results using an adaptive naïve Bayes classifier where source domain consists of labeled data and target domain consists of both labeled and unlabeled data. When we consider target domain labeled data, we split the target domain (movie domain 400 instances) or (DVDs domain 800 instances) or (DVDs domain 400 instances) or (kitchen domain 400 instances) as 100 labeled target domain instances and the remaining as unlabeled target domain instances. The number of test instances are one-third of the total number of unlabeled target domain instances.

2. Case 2: Results using an adaptive naïve Bayes classifier where source domain consists of labeled data and target domain consists of only unlabeled (movie domain consists of 300 instances or DVDs domain consists of 700 instances or kitchen domain consists of 400 instances) data. The number of test instances are one-third of the total number of unlabeled target domain instances. This is considered as the Lower bound for our experiments.

3. Case 3: Results using a supervised classification by using target domain labeled data as training and the one-third of the target domain unlabeled data as testing instances.

4. Case 4: Results using a supervised classification with two-thirds of target domain labeled data along with target domain unlabeled data as training instances and one-third of target domain unlabeled data as testing instances. This is considered as the Upper bound for our experiments.

In the following tables we use represent the movie domain as M, DVDs domain with 400 instances as D, DVDs with 800 instances as D', kitchen domain as K. We also represent Case 1 as "1", Case 2 as "2", Case 3 as "3" and Case 4 as "4".

5.3.1 Unigrams with Leaf Nodes as Grams

Experiments with Source D' and Target M: The following experiments are performed to decide the number of grams to use as features from the target domain and the number of generalized features we have to use for the source domain. First, we have experimented using the source domain as DVDs (800 instances) and target domain as movie (400 instances). Here we performed all the above four cases considering all the grams, or grams with count >1 or count >2 or count >3 and by taking 50 or 100 as generalized features for the source domain. Next, we have performed all the above specified cases with variable amount of grams by considering the source domain as movie (400 instances) and target domain as DVDs (800 instances) and also by considering source domain as movie (400 instances) and target domain as DVDs (400 instances). FCE represents frequently co-occurring entropy.

The results from Table 5.6, show that Case 4>Case 1>Case 3>Case 2. The results obtained by using the "grams>1", "grams>2" are better than or very close to those obtained using "all the grams" as features. We have also seen that the results obtained by considering the top 100 FCE features as generalized features are generally better than the results obtained using the top 50 FCE features as generalized features.

The results from Table 5.7, show that Case 4>Case 1>Case 3>Case 2. The results obtained by using the "grams>1", "grams>2" are better than using "all the grams" as features. We have also seen there is an increase in the performance of the classifiers by considering the top 100 FCE features as generalized features.

The results from Table 5.8 show that Case 4>Case 1>Case 3>Case 2. The results obtained by using the "grams>1" or "grams>2" along with top 100 FCE features as generalized features are better than using "all the grams" as features. By looking at the results from Table 5.8 and 5.7, we have also observed that the performance of the classifier is decreased when we have more amount of unlabeled data in the target domain. It might be because the source domain is not having sufficient amount of labeled data to train a good classifier.

Table 5.6: *Results using unigrams with leaf nodes as grams for Source D' and Target M*

Grams	FCE	Case	Positive				Negative			
			Precision	Recall	F1	ROC	Precision	Recall	F1	ROC
all	50	1	0.5573	0.8516	0.6733	0.7136	0.6793	0.3170	0.4256	0.7136
		2	0.5113	0.3703	0.4170	0.5363	0.5106	0.6506	0.5663	0.5363
		3	0.7933	0.4666	0.5836	0.7230	0.7053	0.8656	0.7203	0.7230
		4	0.7546	0.8206	0.7860	0.8636	0.8026	0.7330	0.7663	0.8636
	100	1	0.7236	0.6973	0.7076	0.7823	0.7196	0.7386	0.7270	0.7733
		2	0.3250	0.0613	0.1020	0.4383	0.4840	0.8793	0.6240	0.4383
		3	0.7933	0.4666	0.5836	0.7230	0.7053	0.8656	0.7203	0.7230
		4	0.7546	0.8206	0.7860	0.8636	0.8026	0.7330	0.7663	0.8636
>1	50	1	0.6603	0.8076	0.7253	0.7753	0.7710	0.5843	0.6593	0.7753
		2	0.5150	0.8500	0.6463	0.5833	0.4190	0.2273	0.2833	0.5836
		3	0.7563	0.4656	0.5710	0.7380	0.6150	0.8493	0.7120	0.7380
		4	0.7923	0.8293	0.8060	0.8636	0.8200	0.7793	0.7963	0.8636
	100	1	0.6760	0.7400	0.7060	0.7740	0.7140	0.6473	0.6780	0.7740
		2	0.6283	0.6326	0.6253	0.6773	0.6186	0.6066	0.6073	0.6773
		3	0.7563	0.4656	0.5710	0.7380	0.6150	0.8493	0.7120	0.7380
		4	0.7923	0.8293	0.8060	0.8636	0.8200	0.7793	0.7963	0.8636
>2	50	1	0.6536	0.6756	0.6620	0.7253	0.6576	0.6413	0.6470	0.7253
		2	0.5030	0.8730	0.6282	0.5090	0.6150	0.1340	0.1800	0.5090
		3	0.7236	0.4956	0.5850	0.7203	0.6080	0.8003	0.6883	0.7203
		4	0.7953	0.8110	0.8023	0.8670	0.7973	0.7890	0.7926	0.8670
	100	1	0.6800	0.6550	0.6660	0.7460	0.6600	0.6880	0.6730	0.7460
		2	0.6840	0.6193	0.6453	0.6810	0.6506	0.7176	0.6780	0.6810
		3	0.7236	0.4956	0.5850	0.7203	0.6080	0.8003	0.6883	0.7203
		4	0.7953	0.8110	0.8023	0.8670	0.7973	0.7890	0.7926	0.8670
>3	50	1	0.7173	0.7390	0.7260	0.7820	0.7330	0.7083	0.7183	0.7820
		2	0.5560	0.4996	0.5180	0.5803	0.5556	0.6130	0.5773	0.5803
		3	0.7713	0.5846	0.6640	0.7570	0.6670	0.8253	0.7370	0.7570
		4	0.7543	0.8006	0.7763	0.8520	0.7873	0.7410	0.7626	0.8520
	100	1	0.6356	0.6973	0.6646	0.7350	0.6656	0.5980	0.6286	0.7350
		2	0.4840	0.6360	0.5390	0.5360	0.5160	0.3533	0.3970	0.5360
		3	0.7713	0.5846	0.6640	0.7570	0.6670	0.8253	0.7370	0.7570
		4	0.7543	0.8006	0.7763	0.8520	0.7873	0.7410	0.7626	0.8520

Table 5.7: *Results using unigrams with leaf nodes as grams for Source M and Target D′*

Grams	FCE	Case	Positive				Negative			
			Precision	Recall	F1	ROC	Precision	Recall	F1	ROC
all	50	1	0.5833	0.8286	0.6723	0.6490	0.6820	0.3816	0.4350	0.6490
		2	0.4930	0.8246	0.6140	0.4280	0.4013	0.1530	0.1900	0.4206
		3	0.7213	0.3746	0.4523	0.7190	0.5776	0.8323	0.6683	0.7190
		4	0.7360	0.7010	0.7170	0.8150	0.7133	0.7493	0.7296	0.8150
	100	1	0.5066	0.9516	0.6606	0.5156	0.6090	0.0743	0.1310	0.5156
		2	0.4960	0.9546	0.6520	0.4096	0.2326	0.0339	0.0573	0.3940
		3	0.7213	0.3746	0.4523	0.7190	0.5776	0.8323	0.6683	0.7190
		4	0.7360	0.7010	0.7170	0.8150	0.7133	0.7493	0.7296	0.8150
>1	50	1	0.7040	0.4646	0.5593	0.7363	0.6026	0.8076	0.6903	0.7430
		2	0.5213	0.4556	0.4413	0.5170	0.4740	0.5500	0.4750	0.5170
		3	0.7383	0.3850	0.5043	0.7340	0.5820	0.8596	0.6926	0.7340
		4	0.7650	0.7306	0.7466	0.8270	0.7443	0.7793	0.7603	0.8270
	100	1	0.6803	0.6150	0.6420	0.7366	0.6423	0.6966	0.6660	0.7366
		2	0.6930	0.6163	0.6346	0.7280	0.6503	0.6970	0.6563	0.7280
		3	0.7383	0.3850	0.5043	0.7340	0.5820	0.8596	0.6926	0.7340
		4	0.7650	0.7306	0.7466	0.8270	0.7443	0.7793	0.7603	0.8270
>2	50	1	0.7190	0.4806	0.5746	0.7293	0.6103	0.8116	0.6956	0.7293
		2	0.5560	0.4080	0.4520	0.5256	0.4916	0.6236	0.5386	0.5256
		3	0.7330	0.4443	0.5523	0.7226	0.6003	0.8366	0.6986	0.7226
		4	0.7640	0.7296	0.7460	0.8156	0.7393	0.7733	0.7560	0.8156
	100	1	0.6773	0.6383	0.6516	0.7390	0.6626	0.6896	0.6706	0.7390
		2	0.6876	0.4520	0.5430	0.7060	0.5926	0.7936	0.6780	0.7060
		3	0.7330	0.4443	0.5523	0.7226	0.6003	0.8366	0.6986	0.7226
		4	0.7640	0.7296	0.7460	0.8156	0.7393	0.7733	0.7560	0.8156
>3	50	1	0.6780	0.4683	0.5530	0.6983	0.5916	0.7730	0.6706	0.6983
		2	0.4713	0.6300	0.5330	0.4380	0.4716	0.3076	0.3540	0.4380
		3	0.7213	0.4070	0.5110	0.7276	0.5873	0.8360	0.6886	0.7276
		4	0.7906	0.7220	0.7533	0.8346	0.7450	0.8046	0.7726	0.8346
	100	1	0.6753	0.5050	0.5773	0.7083	0.6053	0.7566	0.6720	0.7083
		2	0.6323	0.5370	0.5740	0.6926	0.6023	0.6856	0.6363	0.6926
		3	0.7213	0.4070	0.5110	0.7276	0.5873	0.8360	0.6886	0.7276
		4	0.7906	0.7220	0.7533	0.8346	0.7450	0.8046	0.7726	0.8346

Table 5.8: *Results using unigrams with leaf nodes as grams for Source M and Target D*

Grams	FCE	Case	Positive				Negative			
			Precision	Recall	F1	ROC	Precision	Recall	F1	ROC
all	50	1	0.8783	0.8726	0.8740	0.9370	0.8800	0.8806	0.8786	0.9370
		2	0.4883	0.5693	0.5056	0.5153	0.5860	0.4323	0.4356	0.5153
		3	0.8886	0.6363	0.7216	0.8823	0.7420	0.9126	0.8086	0.8823
		4	0.9000	0.8993	0.8990	0.9713	0.9000	0.9020	0.9003	0.9713
	100	1	0.6640	0.9596	0.7843	0.8996	0.9300	0.5133	0.6593	0.8996
		2	0.5406	0.9726	0.6950	0.5827	0.8523	0.1703	0.2756	0.7693
		3	0.8886	0.6363	0.7216	0.8823	0.7420	0.9126	0.8086	0.8823
		4	0.9000	0.8993	0.8990	0.9713	0.9000	0.9020	0.9003	0.9713
>1	50	1	0.9063	0.8396	0.8713	0.9410	0.8503	0.9140	0.8810	0.9410
		2	0.3970	0.3930	0.3693	0.4896	0.5520	0.5976	0.5440	0.4896
		3	0.8306	0.7136	0.7673	0.8790	0.7483	0.8543	0.7966	0.8790
		4	0.9110	0.8990	0.9030	0.9683	0.9020	0.9143	0.9080	0.9683
	100	1	0.8750	0.8646	0.8686	0.9430	0.8686	0.8733	0.8700	0.9430
		2	0.5353	0.7066	0.6016	0.6300	0.7316	0.4680	0.5460	0.6300
		3	0.8306	0.7136	0.7673	0.8790	0.7483	0.8543	0.7966	0.8790
		4	0.9110	0.8990	0.9030	0.9683	0.9020	0.9143	0.9080	0.9683
>2	50	1	0.9036	0.8070	0.8526	0.9470	0.8246	0.9133	0.8666	0.9470
		2	0.4050	0.3400	0.3916	0.3853	0.4420	0.5180	0.4766	0.3853
		3	0.8753	0.7013	0.7786	0.8920	0.7496	0.8990	0.8173	0.8920
		4	0.9300	0.9400	0.9093	0.9703	0.8866	0.9340	0.9093	0.9703
	100	1	0.9123	0.8266	0.8673	0.9533	0.8410	0.9420	0.8786	0.9533
		2	0.8626	0.8800	0.8713	0.9346	0.8776	0.8603	0.8686	0.9346
		3	0.8753	0.7013	0.7786	0.8920	0.7496	0.8990	0.8173	0.8920
		4	0.9300	0.9400	0.9093	0.9703	0.8866	0.9340	0.9093	0.9703
>3	50	1	0.9313	0.7886	0.8540	0.9473	0.8136	0.9386	0.8713	0.9473
		2	0.3450	0.2660	0.2936	0.3536	0.4806	0.6090	0.5260	0.3536
		3	0.8616	0.7090	0.7780	0.8853	0.7506	0.8846	0.8120	0.8853
		4	0.9463	0.8666	0.9036	0.9630	0.8766	0.9456	0.9090	0.9630
	100	1	0.9263	0.8163	0.8673	0.9536	0.8323	0.9316	0.8786	0.9536
		2	0.8923	0.8553	0.8730	0.9530	0.8586	0.8906	0.8730	0.9580
		3	0.8616	0.7090	0.7780	0.8853	0.7506	0.8846	0.8120	0.8853
		4	0.9463	0.8666	0.9036	0.9630	0.8766	0.9456	0.9090	0.9630

From the above experiments, we have seen that Case 1 results are better than Case 2 results. It is because in Case 1, we are using target domain labeled data along with source domain labeled data for training a classifier to predict the labels for target domain unlabeled data. We have also seen that the results obtained using the grams>1 and also by using grams>2 along with top 50 or 100 FCE features as generalized features are better than any other combinations. Next, we have seen that the results obtained by using source as DVDs (800) and target as movie (400) are better than less amount source domain data such as using source as movie (400) and target as DVDs (800) domain. Case 4 results are considered as upper bound for our experiments. For the above experiments, we have found that the results for Case 4>Case 1>Case 3>Case 2 for majority of the experiments. Thus, from the above experiments we have decided that we will perform the remaining experiments by using grams>1 along with top 100 FCE features as generalized features in source domain and the whole vocabulary as features in target domain.

5.4 Experiment 4 Results

5.4.1 Results Using Unigrams with Leaf Nodes as Grams

In addition to the four different cases, we are performing another experiment (i.e., Case 5) with source domain data as training data and target domain unlabeled data as test data. The target domain unlabeled data is divided into three folds and the results are calculated as the average of the 3 folds. This is considered as the Lower bound for our experiments. We represent Case 5 as "5". Table 5.9 shows that the results for Case 4>Case 1>Case 3>Case 2 >Case 5. From Table 5.9, we can see that sentiment classification across $M \rightarrow K$ and $K \rightarrow D$ domains are better when compared with $K \rightarrow M$ and $D \rightarrow K$.

5.4.2 Results Using All Grams with Leaf Nodes as Grams

Table 5.10, shows that the results for Case 4>Case 1>Case 3>Case 2>Case 5. From Table 5.10, we can see that sentiment classification across $M \rightarrow D$, $M \rightarrow K$ and $K \rightarrow D$ domains

Table 5.9: *Results using unigrams with leaf nodes as grams*

$S \rightarrow T$	Case	Positive				Negative			
		Precision	Recall	F1	ROC	Precision	Recall	F1	ROC
$D \rightarrow K$	1	0.7420	0.8130	0.7690	0.8666	0.7876	0.7360	0.7563	0.8666
	2	0.6506	0.6473	0.6413	0.7466	0.6560	0.6736	0.6573	0.7466
	3	0.7663	0.5783	0.6570	0.7860	0.6600	0.8340	0.7333	0.7860
	4	0.8240	0.8530	0.8290	0.9240	0.8400	0.8366	0.8323	0.9240
	5	0.6750	0.5823	0.6200	0.7260	0.6306	0.7306	0.6720	0.7260
$K \rightarrow D$	1	0.8943	0.7876	0.8370	0.9416	0.8073	0.9066	0.8536	0.9416
	2	0.8943	0.7876	0.8373	0.9446	0.8066	0.9053	0.8530	0.9446
	3	0.8716	0.7226	0.7873	0.8943	0.7666	0.8910	0.8236	0.8943
	4	0.9250	0.8676	0.8940	0.9706	0.8733	0.9256	0.8976	0.9706
	5	0.7540	0.6583	0.7026	0.7920	0.6953	0.7836	0.7366	0.7920
$K \rightarrow M$	1	0.6836	0.5883	0.6323	0.7103	0.6426	0.7346	0.6856	0.7103
	2	0.5396	0.3413	0.4176	0.5010	0.5213	0.7143	0.6020	0.5010
	3	0.7563	0.4906	0.5890	0.7426	0.6273	0.8450	0.7170	0.7426
	4	0.7486	0.7796	0.7633	0.8603	0.7966	0.7406	0.7540	0.8603
	5	0.5633	0.5213	0.5390	0.5933	0.5540	0.5956	0.5713	0.5933
$M \rightarrow K$	1	0.7513	0.8113	0.7843	0.8553	0.7980	0.7400	0.7676	0.8553
	2	0.5890	0.6936	0.6363	0.6363	0.6250	0.5143	0.5633	0.6363
	3	0.7663	0.5446	0.6350	0.7776	0.6490	0.8323	0.7283	0.7776
	4	0.8343	0.8203	0.8266	0.9116	0.8206	0.8330	0.8260	0.9116
	5	0.5873	0.6396	0.6123	0.6373	0.5976	0.5446	0.5693	0.6373

are better when compared with $D \rightarrow M$, $K \rightarrow M$ and $D \rightarrow K$. From Table 5.9 and Table 5.10, we can also observe that the results obtained using "unigrams with leaf nodes" as features are better than or very near to the results obtained using "all grams with leaf nodes" as features.

5.4.3 Results Using Unigrams without Leaf Nodes as Grams

Table 5.11, shows that the results for Case 4>Case 3>Case 1>Case 2>Case 5 in majority of the cases. By comparing the results from Table 5.11 with the results from Table 5.9 and Table 5.10, we can state that by using "unigrams without leaf nodes" as features decreases the performance of sentiment classification across domains. Thus the results suggests that "unigrams without leaf nodes" are not as informative as "all grams with leaf nodes" and "unigrams with leaf nodes".

5.4.4 Results Using All Unigrams as Grams

Table 5.12 shows that the results for Case 4>Case 1>Case 3>Case 2>Case 5 in majority of the cases. By comparing the results from Table 5.12 with the results from Table 5.9, Table 5.10 and Table 5.11, we observed that using "all unigrams" as features are better than using "unigrams without leaf nodes" as features. However, "all unigrams" are not as informative as "all grams with leaf nodes" and "unigrams with leaf nodes".

5.5 Experiment 5 Results

5.5.1 Results Using All Grams with Leaf Nodes from PT Trees as Grams

Table 5.13 shows that the results for Case 4>Case 1>Case 3>Case 2>Case 5 in majority of the cases. By comparing the results from Table 5.13 with the results from Table 5.10, we observed that using grams derived from "path tree" as features are not as good as the grams derived from the complete syntax tree. However, we have observed that sentiment

Table 5.10: *Results using all grams with leaf nodes as grams*

$S \rightarrow T$	Case	Positive				Negative			
		Precision	Recall	F1	ROC	Precision	Recall	F1	ROC
$M \rightarrow D'$	1	0.6156	0.7286	0.6650	0.6760	0.6736	0.5410	0.5953	0.6760
	2	0.5103	0.8833	0.6443	0.5046	0.4530	0.1473	0.2116	0.5046
	3	0.7433	0.3753	0.4963	0.7163	0.5820	0.8696	0.6966	0.7263
	4	0.7510	0.7160	0.7323	0.8063	0.7316	0.7630	0.7463	0.8063
	5	0.5770	0.7300	0.6440	0.6496	0.6283	0.4630	0.5323	0.6496
$D' \rightarrow M$	1	0.6056	0.6950	0.6453	0.6923	0.6356	0.5363	0.5756	0.6923
	2	0.5950	0.5093	0.5450	0.6013	0.5643	0.6483	0.6003	0.6013
	3	0.8103	0.4570	0.5553	0.6840	0.6143	0.8573	0.7083	0.6840
	4	0.7756	0.7480	0.7580	0.8383	0.7590	0.7876	0.7703	0.8383
	5	0.6143	0.4580	0.5246	0.6086	0.5726	0.7206	0.6376	0.6086
$M \rightarrow D$	1	0.8880	0.8866	0.8863	0.9443	0.8870	0.8870	0.8866	0.9443
	2	0.5933	0.8336	0.6926	0.7060	0.7653	0.4393	0.5536	0.7060
	3	0.8760	0.6620	0.7466	0.8866	0.7303	0.8986	0.8030	0.8866
	4	0.9426	0.8733	0.9063	0.9790	0.8820	0.9473	0.9130	0.9790
	5	0.5313	0.6146	0.5693	0.5676	0.5433	0.4590	0.4970	0.5676
$D \rightarrow K$	1	0.7643	0.6786	0.7136	0.8003	0.7120	0.7906	0.7446	0.8003
	2	0.6463	0.5646	0.5976	0.6953	0.6140	0.6906	0.6460	0.6953
	3	0.7503	0.5260	0.6160	0.7280	0.6373	0.8286	0.7190	0.7280
	4	0.8320	0.7850	0.8036	0.8943	0.7980	0.8433	0.8160	0.8943
	5	0.7060	0.5630	0.6243	0.7160	0.6376	0.7676	0.6946	0.7160
$K \rightarrow D$	1	0.9403	0.8393	0.8866	0.9666	0.8553	0.9463	0.8986	0.9666
	2	0.9150	0.8520	0.8813	0.9540	0.8980	0.9193	0.8906	0.9540
	3	0.8456	0.6896	0.7573	0.8970	0.7380	0.8736	0.7983	0.8970
	4	0.9386	0.9006	0.9190	0.9820	0.9030	0.9386	0.9203	0.9820
	5	0.7940	0.6930	0.7396	0.8123	0.7276	0.8216	0.7713	0.8123
$K \rightarrow M$	1	0.6196	0.5486	0.5770	0.6396	0.6600	0.6946	0.6373	0.6396
	2	0.4710	0.3326	0.3866	0.4460	0.4906	0.6390	0.5533	0.4446
	3	0.7133	0.4230	0.5253	0.6863	0.5903	0.8283	0.6870	0.6863
	4	0.7580	0.7806	0.7663	0.8350	0.7750	0.7473	0.7593	0.8350
	5	0.5516	0.5286	0.5393	0.5596	0.5410	0.5626	0.5510	0.5596
$M \rightarrow K$	1	0.7723	0.7213	0.7443	0.8316	0.7396	0.7866	0.7610	0.8316
	2	0.5793	0.6610	0.6176	0.6456	0.6066	0.5183	0.5576	0.6456
	3	0.8033	0.5306	0.6243	0.7256	0.6506	0.8563	0.7333	0.7256
	4	0.8316	0.7800	0.8036	0.8910	0.7930	0.8416	0.8153	0.8910
	5	0.5610	0.6206	0.5883	0.6243	0.5753	0.5143	0.5420	0.6243

Table 5.11: *Results using unigrams without leaf nodes as grams*

$S \rightarrow T$	Case	Positive				Negative			
		Precision	Recall	F1	ROC	Precision	Recall	F1	ROC
$M \rightarrow D'$	1	0.5456	0.4853	0.5043	0.5453	0.5196	0.5760	0.5390	0.5453
	2	0.5470	0.5050	0.4966	0.5480	0.5056	0.5473	0.4953	0.5480
	3	0.5943	0.3740	0.4570	0.6113	0.5466	0.7516	0.6323	0.6113
	4	0.6166	0.6203	0.6180	0.6600	0.6180	0.6143	0.6156	0.6600
	5	0.5176	0.6843	0.5893	0.5343	0.5373	0.3626	0.4326	0.5343
$D' \rightarrow M$	1	0.5133	0.4496	0.4753	0.5426	0.4993	0.5626	0.5263	0.5426
	2	0.4776	0.4770	0.4740	0.5033	0.4663	0.4670	0.4633	0.5033
	3	0.6593	0.4713	0.5416	0.6686	0.5910	0.7533	0.6590	0.6686
	4	0.6793	0.6743	0.6766	0.7356	0.6726	0.6776	0.6750	0.7356
	5	0.5243	0.5216	0.5220	0.5310	0.5243	0.5273	0.5246	0.5310
$M \rightarrow D$	1	0.5223	0.4540	0.4813	0.5470	0.5130	0.5786	0.5403	0.5470
	2	0.5133	0.5610	0.5316	0.4763	0.4943	0.4506	0.4646	0.4583
	3	0.6943	0.5080	0.5803	0.6796	0.6080	0.7660	0.6736	0.6796
	4	0.6540	0.6660	0.6576	0.6950	0.6566	0.6446	0.6483	0.6950
	5	0.4903	0.6020	0.5393	0.4703	0.4843	0.3746	0.4206	0.4703
$M \rightarrow K$	1	0.5470	0.4893	0.5140	0.5510	0.5340	0.5896	0.5586	0.5510
	2	0.4576	0.4900	0.4693	0.4593	0.4493	0.4173	0.4286	0.4593
	3	0.6696	0.4430	0.5306	0.6680	0.5813	0.7786	0.6636	0.6680
	4	0.6180	0.6203	0.6110	0.6803	0.6176	0.6140	0.6080	0.6803
	5	0.5066	0.5860	0.5426	0.5483	0.5176	0.4350	0.4723	0.5483
$K \rightarrow M$	1	0.4736	0.4196	0.4416	0.5200	0.4760	0.5280	0.4976	0.5200
	2	0.4536	0.4236	0.4363	0.4750	0.4600	0.4896	0.4723	0.4750
	3	0.6513	0.4390	0.5203	0.6636	0.5760	0.7650	0.6540	0.6636
	4	0.6436	0.6750	0.6530	0.7400	0.6580	0.6316	0.6390	0.7400
	5	0.4800	0.5056	0.4873	0.4936	0.4803	0.4546	0.4613	0.4936
$K \rightarrow D$	1	0.5630	0.4916	0.5246	0.5803	0.5506	0.6210	0.5836	0.5803
	2	0.5026	0.4323	0.4643	0.5313	0.5020	0.5720	0.5343	0.5313
	3	0.6806	0.4850	0.5650	0.6800	0.6010	0.7726	0.6760	0.6800
	4	0.6696	0.6246	0.6463	0.7186	0.6510	0.6953	0.6723	0.7186
	5	0.6166	0.5930	0.6033	0.6343	0.6063	0.6286	0.6163	0.6343
$D \rightarrow K$	1	0.5716	0.4236	0.4853	0.5800	0.5436	0.6856	0.6053	0.5800
	2	0.5113	0.4106	0.4493	0.5040	0.4930	0.5890	0.5330	0.5040
	3	0.7046	0.4850	0.5393	0.6890	0.5953	0.8150	0.6860	0.6890
	4	0.6690	0.6566	0.6580	0.7393	0.6613	0.6700	0.6606	0.7393
	5	0.5703	0.5763	0.5700	0.6000	0.5686	0.5616	0.5620	0.6000

Table 5.12: *Results using all unigrams as grams*

$S \rightarrow T$	Case	Positive				Negative			
		Precision	Recall	F1	ROC	Precision	Recall	F1	ROC
$M \rightarrow D'$	1	0.6526	0.3173	0.4213	0.6550	0.5516	0.8343	0.6630	0.6550
	2	0.5526	0.4363	0.4563	0.5826	0.5310	0.6356	0.5636	0.5813
	3	0.7790	0.2776	0.3900	0.7206	0.5633	0.9203	0.6970	0.7206
	4	0.7583	0.7056	0.7303	0.8173	0.7253	0.7743	0.7486	0.8173
	5	0.5340	0.6340	0.5793	0.5666	0.5446	0.4430	0.4883	0.5666
$D' \rightarrow M$	1	0.5230	0.8496	0.6326	0.5873	0.5943	0.2060	0.2346	0.5873
	2	0.4886	0.7410	0.5850	0.5013	0.3300	0.2316	0.2723	0.5210
	3	0.7933	0.4206	0.5383	0.7250	0.6040	0.8780	0.7120	0.7250
	4	0.7806	0.8006	0.7893	0.8683	0.7950	0.7756	0.7833	0.8683
	5	0.5436	0.4513	0.4923	0.5486	0.5316	0.6223	0.5723	0.5486
$M \rightarrow D$	1	0.7376	0.6290	0.6773	0.7823	0.6736	0.7743	0.7190	0.7823
	2	0.4523	0.5200	0.4836	0.3936	0.4083	0.3536	0.3783	0.3936
	3	0.9100	0.5036	0.6466	0.8570	0.6590	0.9543	0.7790	0.8570
	4	0.9166	0.8566	0.8836	0.9563	0.8623	0.9220	0.8890	0.9563
	5	0.4936	0.5323	0.5113	0.5203	0.4930	0.4546	0.4723	0.5203
$K \rightarrow D$	1	0.8756	0.7713	0.8196	0.9050	0.7930	0.8826	0.8353	0.9050
	2	0.8033	0.7600	0.7853	0.8783	0.7740	0.8103	0.7913	0.8783
	3	0.8736	0.4836	0.6170	0.8306	0.6416	0.9250	0.7550	0.8306
	4	0.9230	0.8306	0.8733	0.9483	0.8406	0.9230	0.8783	0.9483
	5	0.7183	0.6696	0.6923	0.7540	0.6843	0.7246	0.7033	0.7540
$D \rightarrow K$	1	0.6876	0.8013	0.7376	0.7950	0.7640	0.6340	0.6900	0.7950
	2	0.6323	0.5243	0.5706	0.7036	0.5996	0.7006	0.6453	0.7036
	3	0.7920	0.3360	0.4493	0.7506	0.5753	0.8896	0.6943	0.7506
	4	0.8003	0.7836	0.7916	0.8883	0.7936	0.8070	0.8000	0.8883
	5	0.6826	0.5740	0.6233	0.7150	0.6316	0.7326	0.6783	0.7150
$M \rightarrow K$	1	0.6553	0.5513	0.5983	0.6823	0.6110	0.7110	0.6563	0.6823
	2	0.4780	0.5706	0.5163	0.4990	0.4713	0.3786	0.4166	0.4990
	3	0.8193	0.3210	0.4606	0.7263	0.5786	0.9346	0.7136	0.7263
	4	0.8176	0.7886	0.8006	0.8796	0.7936	0.8326	0.8100	0.8796
	5	0.5243	0.5683	0.5423	0.5353	0.5290	0.4846	0.5026	0.5353
$K \rightarrow M$	1	0.5353	0.5443	0.5380	0.5743	0.5426	0.5330	0.5360	0.5743
	2	0.4883	0.6536	0.5573	0.4956	0.4633	0.3096	0.3670	0.4956
	3	0.7233	0.4353	0.5193	0.7130	0.6016	0.8263	0.6913	0.7130
	4	0.7783	0.8206	0.7983	0.8510	0.8120	0.7673	0.7880	0.8510
	5	0.4966	0.4826	0.4880	0.5103	0.5046	0.5196	0.5106	0.5103

classification across $M \rightarrow D$, $M \rightarrow K$ and $K \rightarrow D$ domains are better when compared with $D \rightarrow M$, $K \rightarrow M$ and $D \rightarrow K$ (i.e., similar to the case when we use grams from complete syntax tree as features).

5.5.2 Results Using Unigrams with Leaf Nodes from PT Trees as Grams

Table 5.14 shows that the results for Case 4>Case 1>Case 3>Case 2>Case 5 in majority of the cases. By comparing the results from Table 5.14 with the results from Table 5.9, we observed that using grams derived from "path tree" as features are not as good as the grams derived from the complete syntax tree for performing sentiment classification across domains. However, we have observed that sentiment classification across $M \rightarrow D$, $M \rightarrow K$ and $K \rightarrow D$ domains are better when compared with $D \rightarrow M$, $K \rightarrow M$ and $D \rightarrow K$ (i.e., similar to the case, where we use grams from complete syntax tree as features). We have also observed that the results obtained using "unigrams with leaf nodes" as features are better than or very near to the results obtained using "all grams with leaf nodes" as features.

Table 5.13: *Results using all grams with leaf nodes as grams from PT-SPS*

$S \rightarrow T$	Case	Positive				Negative			
		Precision	Recall	F1	ROC	Precision	Recall	F1	ROC
$M \rightarrow D$	1	0.8160	0.8073	0.8106	0.8846	0.8093	0.8223	0.8146	0.8846
	2	0.5906	0.7406	0.6520	0.6746	0.6656	0.4956	0.5683	0.6746
	3	0.7496	0.6073	0.6636	0.7953	0.6816	0.7960	0.7326	0.7953
	4	0.8936	0.7976	0.8400	0.9416	0.8203	0.8973	0.8550	0.9416
	5	0.5340	0.5633	0.5446	0.5596	0.5396	0.5100	0.5206	0.5596
$D \rightarrow M$	1	0.6530	0.4346	0.5216	0.6753	0.5723	0.7623	0.6536	0.6753
	2	0.4866	0.1623	0.2423	0.5096	0.4990	0.8326	0.6226	0.5096
	3	0.6453	0.6753	0.6566	0.7046	0.6556	0.6250	0.6356	0.7046
	4	0.7740	0.8183	0.7956	0.8740	0.8283	0.7846	0.8056	0.8740
	5	0.5140	0.3436	0.4110	0.4810	0.5073	0.6756	0.5793	0.4810
$M \rightarrow K$	1	0.7356	0.7856	0.7600	0.7996	0.7720	0.7196	0.7450	0.7996
	2	0.5666	0.5043	0.5320	0.5943	0.5450	0.6020	0.5713	0.5943
	3	0.6986	0.6480	0.6723	0.7533	0.6683	0.7170	0.6916	0.7533
	4	0.7890	0.8096	0.7990	0.8683	0.8116	0.7893	0.8000	0.8683
	5	0.5880	0.6016	0.5940	0.6283	0.5936	0.5800	0.5853	0.6283
$K \rightarrow M$	1	0.7113	0.5536	0.6173	0.7500	0.6330	0.7746	0.6930	0.7500
	2	0.5113	0.2313	0.3183	0.5083	0.5036	0.7793	0.6106	0.5083
	3	0.6570	0.6066	0.6280	0.7276	0.6336	0.6860	0.6560	0.7276
	4	0.7510	0.7773	0.7606	0.8256	0.7653	0.7443	0.7516	0.8256
	5	0.5633	0.5003	0.5270	0.5850	0.5440	0.6033	0.5703	0.5850
$D \rightarrow K$	1	0.7303	0.7020	0.7100	0.7743	0.7100	0.7323	0.7156	0.7743
	2	0.6716	0.6846	0.6766	0.7263	0.6773	0.6653	0.6700	0.7263
	3	0.7060	0.6303	0.6640	0.7646	0.6683	0.7400	0.7006	0.7646
	4	0.7976	0.7916	0.7943	0.8643	0.7946	0.8026	0.7983	0.8643
	5	0.6726	0.6590	0.6646	0.7300	0.6733	0.6873	0.6793	0.7300
$K \rightarrow D$	1	0.8603	0.7536	0.8023	0.9200	0.7786	0.8343	0.8250	0.9200
	2	0.9126	0.7246	0.8063	0.9113	0.7676	0.9326	0.8410	0.9113
	3	0.7290	0.6510	0.6850	0.7843	0.6796	0.7516	0.7116	0.7843
	4	0.8456	0.8183	0.8296	0.9286	0.8180	0.8553	0.8343	0.9286
	5	0.7426	0.6226	0.6746	0.7706	0.6740	0.7903	0.7250	0.7706

Table 5.14: *Results using unigrams with leaf nodes as grams from PT-SPS*

$S \rightarrow T$	Case	Positive				Negative			
		Precision	Recall	F1	ROC	Precision	Recall	F1	ROC
$M \rightarrow D$	1	0.8183	0.7856	0.8016	0.9043	0.7953	0.8266	0.8143	0.9043
	2	0.7460	0.7960	0.7586	0.8760	0.7786	0.7013	0.7186	0.876
	3	0.7903	0.7126	0.7480	0.8286	0.7366	0.8100	0.7703	0.8286
	4	0.8533	0.8470	0.8493	0.9436	0.8473	0.8546	0.8503	0.9436
	5	0.5550	0.5750	0.5640	0.5803	0.5600	0.5400	0.5490	0.5803
$D \rightarrow M$	1	0.7063	0.5123	0.5923	0.6840	0.6156	0.7836	0.6883	0.6840
	2	0.4666	0.2083	0.2783	0.5110	0.4930	0.7693	0.5980	0.5110
	3	0.6743	0.7236	0.6900	0.7463	0.7006	0.7846	0.6580	0.7463
	4	0.7490	0.7986	0.7730	0.8286	0.7863	0.7340	0.7730	0.8286
	5	0.5690	0.3460	0.4296	0.5023	0.5283	0.7346	0.6140	0.5023
$M \rightarrow K$	1	0.7353	0.7943	0.7633	0.8553	0.7736	0.7116	0.7410	0.8553
	2	0.6700	0.4963	0.5833	0.6730	0.6150	0.7480	0.6733	0.6730
	3	0.7113	0.7223	0.7153	0.7826	0.7176	0.7070	0.7106	0.7826
	4	0.7706	0.8533	0.8096	0.8800	0.8016	0.7463	0.7883	0.8800
	5	0.6136	0.5936	0.6033	0.682	0.6053	0.6260	0.6153	0.6820
$K \rightarrow M$	1	0.6883	0.6163	0.6483	0.7620	0.6626	0.7273	0.6923	0.7620
	2	0.6126	0.1993	0.2903	0.5473	0.5250	0.8786	0.6566	0.5473
	3	0.6756	0.7046	0.6866	0.7476	0.6920	0.6506	0.6650	0.7476
	4	0.7606	0.8196	0.7876	0.8430	0.8080	0.7396	0.7703	0.8430
	5	0.6290	0.5533	0.5883	0.6426	0.6013	0.6733	0.635	0.6426
$K \rightarrow D$	1	0.8276	0.7493	0.7863	0.9090	0.7736	0.8453	0.8083	0.9090
	2	0.8273	0.8063	0.8163	0.9096	0.8103	0.8330	0.8213	0.9096
	3	0.7696	0.6800	0.7206	0.8260	0.7200	0.7990	0.7563	0.8260
	4	0.8613	0.8310	0.8453	0.9506	0.8386	0.8643	0.8510	0.9506
	5	0.7393	0.6443	0.6886	0.7836	0.6833	0.7713	0.7243	0.7836
$D \rightarrow K$	1	0.6816	0.7600	0.7156	0.8160	0.7270	0.6473	0.6816	0.8160
	2	0.6023	0.7943	0.6830	0.7073	0.6960	0.4763	0.5630	0.7073
	3	0.6920	0.6906	0.6836	0.7763	0.6903	0.6906	0.6823	0.7763
	4	0.8023	0.8476	0.8240	0.9126	0.8356	0.7930	0.8133	0.9126
	5	0.6636	0.6730	0.6676	0.7346	0.6786	0.6696	0.6730	0.7346

Chapter 6

Discussion and Conclusions

In this chapter, we will discuss the questions raised in Section 4.1 using the results reported in Chapter 5. We also draw some conclusions and address some of the limitations that our approach faces.

The research questions that we raised are restated here, for convenience:

- Are the grams extracted from syntax trees, when used as features in a domain specific classifier, comparable in terms of prediction and accuracy to the structured syntax subtrees? Overall, how useful are these features for the sentence-level sentiment classification problem?

  The results from Table 5.1, Table 5.2 and Table 5.5 show that using different types of grams as features is as good as using structured syntax subtrees as features for a domain specific classifier. We performed experiments using different types of grams from complete syntax tree as well as from path trees. Table 5.2 when compared with Table 5.1, shows that SVM performs very well when considering "all grams with leaf nodes" and "unigrams with leaf nodes" as features. Table 5.5 when compared with Table 5.1, shows that naïve Bayes also performs well when considering "all grams with leaf nodes" and "Unigrams with leaf nodes" as features.

- What is the effect of using various gram types described in Section 3.4 (i.e., all grams with leaf nodes, unigrams with leaf nodes, unigrams without leaf nodes, all unigrams)

for training the domain adaptation classifiers? Is it better to use all the features or a reduced set of features? Also, how many features do we need to use as generalized features?

Table 5.6, Table 5.7 and Table 5.8 show that the results are not very good when we use all the grams (of a particular type) as features. One reason for this is because some of those features are not informative for training a cross-domain classifier. However, when we decrease the number of features (by eliminating the grams that occur only 1 time) we observed that there is an improvement in the performance of sentiment classification across domains in each of the four different cases. But, if we continue this process and reduce more the number of features then there may or may not be an increase in the performance of the classifier across domains. In addition to the above observations, we noticed that the classifier performs well in majority of the experiments when we consider including the top 100 FCE features as generalized features when compared to the results obtained by using top 50 features as generalized features.

In our experiments, we ended up using a threshold value of 1 because selecting the grams that occur more than 1 time reduces the number of unimportant features, while at the same time retaining the important features for the sentiment classification task across domains. We have also used 100 generalized features because we have seen that there is an improvement in the results when we consider 100 generalized features instead of 50 generalized features.

- How does the ANB approach perform when using some target domain labeled data versus not using any target domain labeled data? Are the results better when using some target labeled data?

  We have seen that the results are better if we consider including the target domain labeled data for training a classifier. It is because the classifier will have more direct

supervision from target and will be able to learn more accurate information related to the target domain.

- How does the ANB approach perform when compared to supervised NBM classifiers?

  Supervised naïve Bayes multinomial classifiers are considered to be an upper bound for our ANB experiments, when target domain labeled and unlabeled data is used as training labeled data. We have seen that in the majority of our experiments Case 4>Case 1>Case 3>Case 2>Case 5. We have observed that sentiment classification across $M \rightarrow D$, $M \rightarrow K$ and $K \rightarrow D$ domains are better when compared with $D \rightarrow M$, $K \rightarrow M$ and $D \rightarrow K$ domains. We have also observed that the results obtained using "unigrams with leaf nodes" and "all grams with leaf nodes" as features are better than using "unigrams without leaf nodes" and "all unigrams" .

From the results, we can conclude that the ANB classifier increases the performance of sentiment classification across domains especially when we use $M \rightarrow D, M \rightarrow K, K \rightarrow D$ as our source and target domains. We can also conclude that the results obtained using "all grams with leaf nodes" > "unigrams with leaf nodes" > "all unigrams" >"unigrams without leaf nodes", as we have observed in our preliminary results from Table 5.2 and Table 5.5.

However, while our ANB approach performed very well across different domains, there are some limitations that we would like to address in our future work. One of the major limitations is that, the results obtained using grams from path tree are not as good as results obtained using grams from a complete syntax tree as features. The reason for this might be that path trees are eliminating the information that might be informative to train a classifier. We would like to further explore this issue by considering various strategies in our future work.

Chapter 7

Future Work

In this chapter, we discuss some of the improvements and possible future directions for our approach. In order to overcome the limitation of our approach (i.e., the results obtained using grams from path tree are not as good as results obtained using grams from a complete syntax tree), we would like to perform the following experiments in our future work:

1. First, we would like to perform sentiment classification across domains by training a classifier using grams extracted from MCT trees (obtained using sentiment based pruning strategies).

2. Next, perform sentiment classification across domains, by considering grams extracted from PT trees (obtained using adjective based pruning strategies).

3. Next, we would like to train a classifier using grams extracted from MCT trees (obtained using adjective based pruning strategies).

4. In addition to the above experiments, we would like to explore some interesting POS patterns for a given set of sentences. We would like to use these patterns as features and evaluate the performance of ANB on cross-domain sentiment classification.

Bibliography

John Blitzer, Ryan Mcdonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *In EMNLP*, 2006. URL http://john.blitzer.com/papers/emnlp06.pdf.

John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *In ACL*, pages 187–205, 2007.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995. ISSN 0885-6125. URL http://dx.doi.org/10.1007/BF00994018. 10.1007/BF00994018.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407, 1990.

George Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944974.

Xavier Glorot. Domain adaptation for large-scale sentiment classification : A deep learning approach. *Learning*, 1:513–520, 2011. URL http://www.icml-2011.org/papers/342_icmlpaper.pdf.

Ali Harb, Michel Plantié, Gerard Dray, Mathieu Roche, François Trousset, and Pascal Poncelet. Web opinion mining: how to extract opinions from blogs? In *Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology*,

CSTST '08, pages 211–217, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-046-3. doi: 10.1145/1456223.1456269. URL http://doi.acm.org/10.1145/1456223.1456269.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi: 10.1145/1014052.1014073. URL http://doi.acm.org/10.1145/1014052.1014073.

Thorsten Joachims. Svm-light, 2002. URL http://svmlight.joachims.org/.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423–430, 2003.

Shoushan Li and Chengqing Zong. Multi-domain sentiment classification. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 257–260, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1557690.1557765.

Tom Mitchell. *Machine learning*. McGraw-Hill Companies Inc., international edition, 1997.

Alessandro Moschitti. Tree kernels, 2002. URL http://disi.unitn.it/moschitti/.

Ramanathan Narayanan, Bing Liu, and Alok Choudhary. Sentiment analysis of conditional sentences, 2009. URL http://dl.acm.org/citation.cfm?id=1699510.1699534.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 751–760, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772767. URL http://doi.acm.org/10.1145/1772690.1772767.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, 2005. URL http://www.cs.cornell.edu/people/pabo/movie-review-data.

César R Souza. Kernel functions for machine learning applications, March 2010. URL http://crsouza.blogspot.com/2010/03/kernel-functions-for-machine-learning.html.

Songbo Tan, Xueqi Cheng, Yuefen Wang, and Hongbo Xu. Adapting naive bayes to domain adaptation for sentiment analysis. *ADVANCES IN INFORMATION RETRIEVAL PROCEEDINGS*, 5478:337–349, 2009. URL http://www.springerlink.com/index/2461874p846n1523.pdf.

Wei Zhang, Peifeng Li, and Qiaoming Zhu. Sentiment classification based on syntax tree pruning and tree kernel. In *Proceedings of the 2010 Seventh Web Information Systems and Applications Conference*, WISA '10, pages 101–105, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4193-8. doi: 10.1109/WISA.2010.29. URL http://dx.doi.org/10.1109/WISA.2010.29.