A Domain Adaptation Method for Text Classification based on Self-adjusted Training Approach

by

Iván Garrido Márquez

A Dissertation
Submitted to the Program in Computer Science,
Computer Science Department
in partial fulfillment of the requirements for the degree of

Master in Computer Science

at the

National Institute for Astrophysics, Optics and Electronics
August 2013
Tonantzintla, Puebla

Advisors:

Manuel Montes y Gómez, PhD., INAOE
Luis Villaseñor Pineda, PhD., INAOE

A Domain Adaptation Method for Text Classification based on Self-adjusting Training Approach

Iván Garrido Márquez
Computer Science Department
National Institute for Astrophysics, Optics and Electronics

Abstract

Information grows rapidly everyday, most of this information is kept in digital text documents, web-pages, posts on social networks, blogs, e-mails [39], electronic books [17], and scientific publications [28]. Organizing and categorizing all this text information automatically results helpful for many tasks. Supervised learning is the most successful approach for automatic text classification. Supervised learning assumes that the training and test set come from the same distribution. Sometimes there are not labeled data available on the target domain, instead we have a labeled data set from a similar or related domain that we can use as auxiliary domain. Despite domains are similar, their feature space and the distribution are different, hence the performance of a supervised classifier demeans. This situation is called the domain adaptation problem. The domain adaptation algorithms are designed to narrow the gap between the target domain distribution and the auxiliary domain distribution. The semi-supervised technique of self-training allows to iteratively enrich the training test with data from the test set. Using self-training for domain adaptation presents some challenges in the text classification scenario; first, the feature space changes on each iteration because new vocabulary is transferred from the target domain to the training set, second, a way to select the more confidently labeled instances is needed, because adding wrong labeled instances to the training set will affect the model. Many of the methods addressing this problem need user defined parameters like the number of instances selected per iteration or the stop criteria. Tuning these parameters into a real problem is another problem by itself. On this work we propose a self-adjusting training approach method, which is able to adapt itself to the new distributions obtained on a self-training process. This method integrates some strategies to adjust its own settings each iteration. The proposed method obtains good results on the thematic cross-domain text classification task, it reduces the error rate in 65.13% on average from the supervised learning approach on the testing dataset. It also was tested in the cross-domain sentiment analysis, reducing the error rate by 15.62% on average from the supervised learning approach on the testing dataset. The performance obtained in the evaluation of the proposed method is competitive with other state of the art methods.

Resúmen

La información crece a diario rápidamente, mucha de esta información es almacenada como documentos digitales de texto, páginas web, comentarios en redes sociales, blogs, correos electrónicos [39], libros electrónicos [17], y publicaciones científicas [28]. Organizar y categorizar toda esta información de forma automática es muy útil para varias actividades. El aprendizaje supervisado es el enfoque más exitoso para la clasificación automática de textos. En el aprendizaje supervisado se asume que el conjunto de entrenamiento y el conjunto de prueba provienen de la misma distribución. Algunas veces no hay datos etiquetados disponibles del dominio objetivo, en su lugar se cuenta con un conjunto de datos etiquetados de un dominio similar o relacionado que se puede usar como dominio auxiliar. A pesar de que los dominios son similares, su espacio de atributos y distribución son diferentes, entonces el desempeño de un clasificador supervisado disminuye. Esta situación es conocida como el problema de adaptación entre dominios. Los algoritmos de adaptación entre dominios son diseñados para reducir la brecha entre la distribución del dominio objetivo y la distribución del dominio auxiliar. La técnica semi-supervisada conocida como self-training permite enriquecer iterativamente el conjunto de entrenamiento con datos del conjunto de prueba. El usar self-training para adaptación entre dominios presenta algunos retos en el escenario de clasificación de textos; primero, con cada iteración el espacio de atributos cambia debido a que nuevo vocabulario es transferido desde el dominio objetivo al conjunto de entrenamiento, segundo, se necesita de una manera de seleccionar las instancias etiquetadas más confiablemente, porque si se agregan instancias mal etiquetadas al conjunto de entrenamiento el modelo será afectado. Muchos de los métodos que abordan este problema necesitan de parámetros definidos por el usuario, como el número de instancias seleccionadas por iteración o el criterio de paro. Ajustar estos parámetros en un problema real es un problema por sí mismo. En este trabajo se propone un método auto-ajustable, el cual es capaz de adaptarse a las nuevas distribuciones obtenidas en un proceso de self-training. Este integra algunas estrategias para ajustarse en cada iteración. El método propuesto obtuvo buenos resultados en la tarea de clasificación temática de textos entre dominios, redujo la tasa de error en 65.13% en promedio contra el enfoque supervisado en los datos de evaluación. También fue probado en la tarea de

análisis de opinión entre dominios, reduciendo la tasa de error en 15.62% en promedio contra el enfoque supervisado en los datos de evaluación. El desempeño del método obtenido en la evaluación fue competitivo contra otros métodos en el estado del arte.

Agradecimientos

Primeramente, agradezco a mi amada esposa Marisol, por apoyarme y acompañarme aventuradamente en la persecución de mis sueños. Por amarme tanto al compartir conmigo esta experiencia de permitirnos crecer juntos. Por dar a mi lado este nuevo paso en nuestro viaje a través del universo.

Agradezco a mis Padres que siempre han estado para mí, me han dado su apoyo pleno en todas mis decisiones de vida. A mi Madre, de quién muchos dirían soy reflejo, agradezco por sus incansables esfuerzos por apoyarme, por darme siempre ánimos, y mantener mi confianza en alto. Quién con su vida y acciones me ha enseñado la más grande lección de amor. A mi Padre por que ha sido siempre mi ejemplo de fortaleza y voluntad. De quién he aprendido a ser un hombre digno, ético y trabajador.

Agradezco a mis familiares que siempre han confiado en mi y han estado cerca en todo momento. En especial a mi Tía Paty, quién es un ejemplo y motivación para esta senda de vida que estoy recorriendo hoy día.

Agradezco a mis Asesores Manuel Montes y Gómez, y Luis Villaseñor Pineda por haberme dado la confianza de trabajar con ellos y aprender de ellos. Por compartir su muy valioso conocimiento, amplia experiencia, talento e inteligencia conmigo. Les agradezco su sabia guía en mi primer paso en el camino de la investigación, el cual he decidido seguir. También les agradezco su siempre amigable trato, y amables consideraciones para conmigo.

Agradezco a mis Sinodales, Claudia Feregrino Uribe, Jesús Ariel Carrasco Ochoa, y Carlos Alberto Reyes García por sus acertados comentarios y observaciones. Por sus consejos y tiempo que sin duda ayudaron a mi trabajo a ser de una mejor calidad.

Agradezco al Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE) por admitirme tras sus puertas, y por brindar todas las facilidades necesarias para concluir este trabajo.

Agradezco a todos mis Profesores en la maestría, quienes no solo me impartieron preciado conocimiento, sino que también supieron motivarme en el fascinante campo de las Ciencias Computacionales.

Agradezco a mis Compañeros en el INAOE, con quiénes compartí tremendas desveladas tras agotadoras jornadas; así como también momentos de alegría y convivencia. Con quienes tengo una sincera amistad. Metzli, Daniel, Paco, Ricardo, Roberto y muy especialmente a Ulises y a Lindsey, una de las mejores personas que he conocido en mi vida.

Finalmente, agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo para mis estudios y para la realización de este trabajo por medio de la Beca No. 322606.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Parallel to the increment in human knowledge, the information on the web grows rapidly everyday; in addition, huge amounts of information are generated by many organizations. Nowadays, most of this information is kept in digital text documents[39][17][28]. The costs of having all this text information classified and organized become very high as the amount of information grows. The time to manually complete these tasks for many databases would be just unfeasible, even with the employment of many people, which raises the monetary cost. Another problem is that depending on the contents of the documents, it might require a certain level of expertise on the subjects to do the task. Because of the large size of the document sets, their variety of subjects and types, it becomes highly important to organize and categorize them automatically.

Text categorization has plenty of useful applications on the daily life, indexing journal articles by subject categories, archiving patents with international patent classification, exploiting information from patient records, e-mail filtering, news tracking by topics, hold web pages organized into category hierarchies. Text categorization also gives the advantage of making the access to information easier to the user.

Currently, by using machine learning techniques, the performance of automatic text classification systems is near to the performance obtained by

human experts. The most successful approach for automatic text classification is supervised learning. In supervised learning a model is built by receiving a training set, a set of examples classified by human experts. This model intends to emulate the subjective criteria used by the human experts to classify the test set, a set of new unclassified examples.

The successful performance of supervised learning relies on two assumptions. First, the examples in the test set can be represented in the same feature space as the training set. Second, the features in the test set have the same distribution as the training set.

Nevertheless, there are some circumstances where the examples of the training set do not represent accurately the distribution and the features of the test set, this results in learning an inadequate model for the problem, consequently the performance of the classification demeans.

Lets take as an example an anti-spam system; in the year 2012, about 144 billion emails were sent daily around the world, 68.8% were spam [35]. With a reasonable set of example e-mails labeled as spam or not-spam, a classifier trained with this data might perform well, but after some time the performance will decrease because the new e-mails come from a different distribution. While in the former data the main topics in spam could have been stocks and replicas, the current spam perhaps is talking about pharmaceuticals and dating.

Another example can be found on the need of feedback about products and services the companies offer, based on customer reviews. There is abundant information in blogs, social networks and review web-sites, however gathering a complete set of a particular domain remains as a hard task. Imagine we have a set of movie reviews labeled as positive or negative according to the opinion expressed on them, but we want to classify as positive or negative a set of reviews of books, even though a model of positive and negative opinions can be built from the movie data, it will not necessarily be able to classify book reviews with the same precision, since the words used to describe positive and negative opinions in the movies domain are not the same in the books domain, so the feature space is different.

1.2 The Problem

On December of 2012, the number of websites in the world reached 634 millions, with a growth of 51 millions of sites that year. An average of 175 million tweets were sent daily in 2012 [35]. Analogous to the fast increase of information and the number documents containing it, the need for algorithms that can automatically classify those documents rises. The evolution of information makes hard and expensive the task of generating large labeled data sets.

Generating a large enough labeled data set, to appropriately represent a domain of knowledge as a training set in a supervised learning approach, requires the gathering of a great amount of data. In the case of text classification problems to gather a training set we need multiple text documents, about the domain of interest (target domain). The aid of human experts on the domain is required for reading and correctly labeling the training data. The manual labeling process might last for long, since it depends on the number of documents available, the human experts available, and the length of the documents. Another consideration could be the subjectivity of the criteria, used to classify the text documents by the human experts.

The following situations can be identified, where a training set might not represent well enough the target domain:

1. There only exist very few labeled examples of the target domain, the model trained with these data will be poor and imprecise.

2. The topics of the domain change swiftly and constantly, the model trained with the gathered data will become obsolete soon.

3. There does not exist any labeled example on the target domain, but it is possible to get a labeled data set from a similar domain; although the domains are similar, the features to characterize the examples and the distributions may differ, from one domain to another.

4. In spite of the huge amount of labeled data from the target domain, they are all written in a different language than the one of the test set, in this case a different vocabulary means a different feature set, also cultural and regional elements lead to a different distribution.

As a result of the previous situations, labeled data sets are meager or completely nonexistent in multiple domains. It becomes desirable to be able to exploit the information in a data set from a similar domain, where a considerable amount of labeled data is available; moreover it is also desirable to exploit that information along with any information that can be gotten from the target domain.

With the objective of handling the situations where little or none examples are available in the target domain, an alternative is the use of a set of labeled data from a related domain to learn a classifier, from this point we will refer this related domain as the auxiliary domain. Then by taking advantage of any information we can get from the unlabeled data of the target domain, we can make the distribution learned by the classifier closer to the target domain distribution.

1.3 Objectives

1.3.1 Main Objective

Proposing and evaluating a domain adaptation method, for text classification problems, based on a dynamic instance selection criterion during a self-training approach.

1.3.2 Specific Objectives

1. To design a strategy for integrating confident labeled examples of the target domain into the training set to build a more accurate model.

2. To identify a reliable stop criterion that allows the method to achieve fair results in a limited number of iterations.

3. To analyze a relation criteria between the domains, for evaluating the classification problem and to be able of determining the ability of the proposed method to solve such problem.

4. To evaluate the proposed method in a thematic cross-domain classification scenario and a non-thematic cross-domain classification scenario.

1.4 Overview of the document

The rest of this document is organized as follows. Chapter 2 contains theoretical information, such as useful concepts and definitions to better understand the next chapters. Chapter 3 describes some of the previous related work on the same problem, it also presents a taxonomy by the most prevalent approaches identified. Chapter 4 explains in detail the methods proposed on this work as a solution for the domain adaptation problem in text classification. Chapter 5 shows the experimental environment, design, testing tasks and datasets. Chapter 6 depicts and discusses the achieved results in the evaluation of the proposed solutions. Finally Chapter 7 enumerates the conclusions on the observed results, additionally it glimpses the future work we are interested in.

Chapter 2

Background information

This Chapter explains the theoretical bases of the concepts taken for the development of this work. The next section defines the text categorization problem in a general way. The rest of the chapter describes the details of how data is represented, the machine learning algorithms used in the proposed method, and how to evaluate the performance of text categorization. The chapter closes with an overview of semi-supervised learning which was the approach adopted for developing our method.

2.1 Text categorization

In the text categorization problem, the objective is to assign automatically a category label or labels to a set of unlabeled text documents, from a predefined set of categories. Currently, due to a combination of information retrieval, and machine learning techniques, the effectiveness of automatic text categorization systems competes with the one obtained by classifying text documents with the help of human experts.

The mainly applied technique for text categorization is the supervised learning. This technique attempts to reproduce the subjective criteria used by human experts, to classify a collection of examples or instances from a particular classification problem. The task of text categorization with a supervised classification method is described next.

Given a set $D = \{d_1, d_2, d_3, ..., d_m\}$ of text documents, and a set of categories or classes $C = \{c_1, c_2, ..., c_n\}$, the text categorization task consists on obtaining a function f', called the classifier, that approximates an unknown function $f : D \rightarrow C$, which describes how an expert would label each document in a class. For supervised classification there is a known set of labeled data represented by a set of tuples $D_t = \{(d_1, c_1), ..., (d_x, c_x)\}$ called the training set. Each document $d_i \in D_t$ is described by a vector of features or attributes from a feature space $W = \{w_1, w_2, w_3, ...w_y\}$, commonly the words or terms appearing in the texts. Using a learning algorithm a model is obtained by analyzing the distribution of the data in the training set, this means that a function $f'(D) = C$ is chosen to predict the category labels of future unlabeled documents. If the learned model fits the distribution of the training set, whenever it is evaluated with new unlabeled data coming out from the learned distribution, the model will perform predicting well the category labels of the new data. A validation set could be used for tunning classification parameters in order to avoid overfitting [38].

As we mentioned above, a text document is represented as a vector of words. The next section explains a common way for representing text documents for supervised classification.

2.2 Bag of words

Although there are many different representations of the text documents, the Bag of Words (BoW) remains as one of the most popular in text categorization and information retrieval tasks. In the BoW model, every document is represented as an unordered collection of the words that occur in it.

Each document becomes a vector of features. The feature space is formed by all the different words contained in the collection of documents. All the features are represented as a dimension of a vector space. In text classification the feature space normally is highly-dimensional.

Every feature in a vector takes a value according to the presence and importance of the word in the document. There are several ways for weighting the terms. In a binary weighting, a feature will get a value of 1 if the term exists in the document, and 0 if it does not. Another way is the term

frequency or tf, when the value of a feature is the number of times the term occurs inside the document. A way to weight the terms, according to its importance in both the documents and the corpus, is $tf - idf$, df is the number of documents in the corpus where the term appears.

$$idf_j = \log\left(\frac{n}{df_j}\right) \tag{2.1}$$

$$tf - idf_{i,j} = tf_{i,j} \times idf_j \tag{2.2}$$

i stands for the document i and j for the term j. If a term occurs several times inside a document, the weight of this term rises up. On the other hand, if the term occurs among several different documents the term will get its weight decreased [36].

2.3 Supervised learning algorithms

Supervised learning algorithms take class labeled examples as an input and learn from them how to predict the class labels for new unlabeled data. The output of a learning method is a model or a predictive function. As we said on Chapter 1 supervised learning is the most successful approach for automatic text categorization. On this section we will explain the algorithms taken to build the mechanics of the proposed method.

2.3.1 Support vector machines

Support vector machines (SVM) is a supervised learning algorithm, proposed in 1995 by Vapnik [40], based on the structural risk minimization principle from computational learning theory, which tries to find a hypothesis that guarantees the lowest error on an unseen test example. SVM was first designed only to handle binary classification problems [10].

Given a training set X of examples of a binary classification problem, where each example is a vector of d dimensions, $x_i \in \mathbb{R}^d$ with a class $y_i \in Y\{-1, 1\}$, SVM assumes these examples are linearly separable, i.e., there is at least one hyperplane that can separate the two classes. The hyperplane can be described by:

$$w \centerdot x + b = 0 \tag{2.3}$$

The w vector is normal to the hyperplane, $b/\parallel w \parallel$ is the perpendicular distance from the hyperplane to the origin, and $\parallel w \parallel$ is the euclidean norm of w. w and b are parameters controlling the decision rule:

$$x_i.w + b \geq +1 \quad y_i = +1 \tag{2.4}$$

$$x_i.w + b \leq -1 \quad y_i = -1 \tag{2.5}$$

There are many hyperplanes that result in the same classification on the training set, SVM chooses the separation hyperplane with the maximum margin, the margin is the perpendicular distance separating the closest examples of both classes to the hyperplane, these examples are called the support vectors. Choosing the maximum margin hyperplane attempts to increase the ability to classify correctly previously unseen examples.

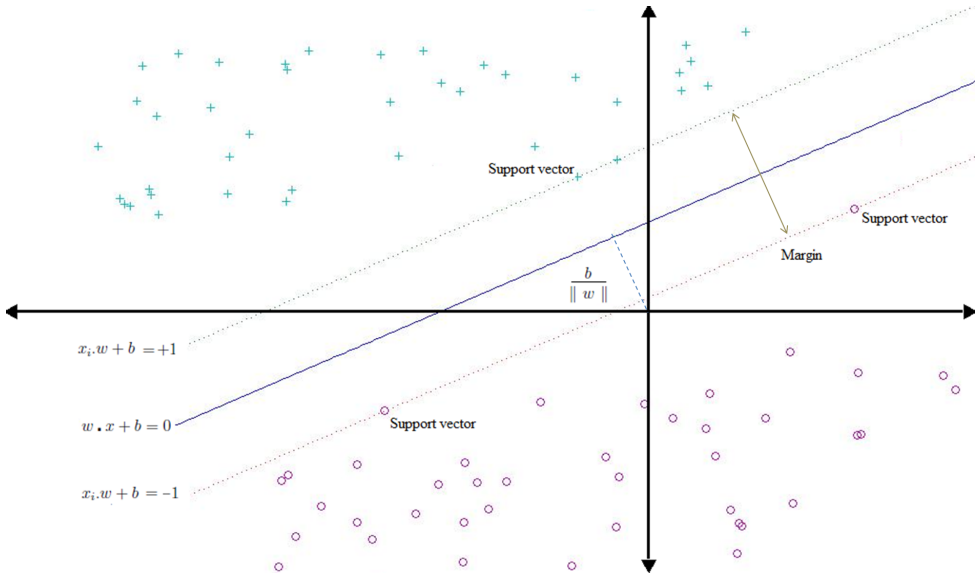

Figure 2.1: Support Vector Machines Diagram.

If the problem is not linearly separable, the SVM algorithm may be modified by adding a softening variable, the idea is to allow some examples passing the hyperplane margins with certain penalty.

When there is not a possible separation linear hyperplane, a solution is to create a non-linear classifier using a kernel function. The kernel function projects the problem from the current dimensional space to a higher dimensional space where the data could be linearly separable. Some popular kernel functions are, Gaussian radial basis function, Polynomial and Hyperbolic tangent [24].

If the problem is not binary, multiple binary classifiers should be constructed. Many strategies have appeared to divide the multiclass data into a binary problem, two of them that are simple and popular are one-versus-all and one-versus-one [19] [32] [31], the first takes each different class against the rest of the other classes, the label is chosen by the highest value from the calibrated output functions; on the other hand the second strategy constructs a classifier for each combination of two classes, the label is chosen by a voting scheme [8].

2.3.2 Prototype classifier

The nearest prototype classifier is one of the most simple learning algorithms. A prototype is one artificial representative instance of each class, built from the training labeled data. The decision rule, for prediction of the class of a new unseen instance is the class of the closest prototype to this new instance. A distance or a similarity measure should be chosen in order to compare how close is a new instance to the different prototypes.

Let p_i be the prototype of class i, x_j an unlabeled instance to be classified, and $sim(x_j, p_i) \in R^n$ a function which measures the similarity or the distance between two vectors described with the same set of attributes [16]. The prototype classifier decision rule $f(x_j)$ may be defined as:

$$f(x_j) = argmax_i(sim(x_j, p_i)) \tag{2.6}$$

One of the most basic prototype construction methods is the arithmetical average of class [18], the prototypes are built by the mean vector of its class. The idea is to identify the center of mass of each class assuming a

homogeneous mass distribution on each example.

$$\overrightarrow{p_c} = \frac{\sum_{d \in D} \overrightarrow{d_k}}{|D|} \tag{2.7}$$

The Rocchio algorithm was originally created for the relevance feedback of a query, on the information retrieval task[37]. On this algorithm all the documents are considered, whether if they are relevant or not, the mean of relevant documents have positive influence, and the non relevant negative [23].

$$\overrightarrow{p_c} = \alpha \frac{1}{|c|} \sum_{d \in c} \overrightarrow{d_k} - \beta \frac{1}{|\overline{c}|} \sum_{d \in \overline{c}} \overrightarrow{d_j} \tag{2.8}$$

Another method to construct the prototype is the Cumuli geometric centroid, where each term will be given a summation weight[18].

$$\overrightarrow{p_c} = \sum_{d \in D} \overrightarrow{d_k} \tag{2.9}$$

Cosine similarity

Because of the good directional properties of high dimensional spaces like text, cosine similarity is a popular similarity measure for text documents represented in a vectorial form. The cosine similarity measures the angle between two vectors. If we think of a document as a vector drawn from the origin to the point of the values of all its dimensions, exactly equal documents will overlap to each other, very similar documents will have a small angle between them, while very dissimilar documents will have a wide angle between them. As the values of their dimensions in frequency term representations cannot be less than 0, documents orthogonal to each other are completely dissimilar, because they do not share any term [29] .

By taking the Cosine of this angle, we get values from -1 to 1. As we said the most dissimilar case in a frequency term representation of text is to have orthogonal vectors, that means the greatest angle we could find is 90, so the lower value of the cosine is 0. The cosine of the angle indicates the similarity, the smaller the angle, the cosine will be closer to 1, and the similarity will

be bigger. On the other hand, the wider the angle, the cosine will be closer to 0, and the similarity will be lower.

The calculation of the cosine similarity comes from the dot product of two vectors $a \cdot b = \|a\|\|b\|cos(\theta)$, where θ is the angle between a and b. Therefore the cosine of the angle can be obtained from:

$$cos(\theta) = \frac{a \cdot b}{\|a\|\|b\|} \tag{2.10}$$

2.3.3 Naive Bayes

Given a set X of examples x_i, each represented as a vector of attributes $x_i = (a_1, a_2, ..., a_n)$, the bayesian approach takes advantage of the conditional probability to predict the class label $y_j \in Y$ of the new unlabeled instances.

$$f(x) = Y \tag{2.11}$$

$$f(x) = argmax_{y_j \in Y} P(y_j) P(y_j | a_1, a_2, ..., a_n) \tag{2.12}$$

using the Bayes theorem:

$$f(x) = argmax_{y_j \in Y} P(y_j) \frac{P(a_1, a_2, ..., a_n | y_j) P(y_j)}{P(a_1, a_2, ..., a_n)} \tag{2.13}$$

$$f(x) = argmax_{y_j \in Y} P(y_j) P(a_1, a_2, ..., a_n | y_j) P(y_j) \tag{2.14}$$

$P(y_j)$ can be easily estimated just by counting the appearances of y_j among the training labeled data. In order to estimate $P(a_1, a_2, ..., a_n | y_j)$ in a feasible way, we can assume that all the attribute values are conditionally independent, the probability of a target y_j given some values of $a_1, a_2, ..., a_n$ becomes the product of their individual probabilities $\prod_k P(a_k | y_j)$, the Naive Bayes classifier is defined by:

$$f(x) = argmax_{y_j \in Y} P(y_j) \prod_k P(a_k | y_j) \tag{2.15}$$

2.4 Evaluating the text categorization

Once the classification model or classifier is constructed, its effectiveness could be evaluated by testing it, and measuring the performance of the results on a testing set. The effectiveness of a classifier is normally evaluated by the intuitive measure of the accuracy, the percentage of correct category label predictions.

$$accuracy = \frac{number\ of\ correct\ predictions}{total\ of\ predictions} \tag{2.16}$$

The error is $1 - accuracy$. Whenever the examples of a certain class outnumbers by far the examples in each of the other classes, this is called an unbalanced classes problem, if the classifier always predicts this class, the accuracy could be very high regardless the real predictive effectiveness; because of this problem some other measures have been proposed for evaluating classification performance.

Precision measures the percentage of correctly classified examples of a certain class, among all the examples classified in that class. Recall measures the percentage of correctly classified examples of a certain class, among all the examples that actually belong to that class.

As both precision and recall are measured per class, a way to average the results for each individual class is needed, this can be done in two different ways, the first one is micro-averaging, counting each class proportionally to the number of its examples in the training set, the second one, macro-averaging, is considering all the classes counting the same, without matter how many examples of each class are in the training set.

$$Micro - average\ Precision = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} TP_i + FP_i} \tag{2.17}$$

$$Macro - average\ Precision = \frac{\sum_{i=1}^{|C|} \frac{TP_i}{TP_i + FP_i}}{|C|} \tag{2.18}$$

$$Micro - average\ Recall = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} TP_i + FN_i} \tag{2.19}$$

$$Macro - average\ Recall = \frac{\sum_{i=1}^{|C|} \frac{TP_i}{TP_i + FN_i}}{|C|} \tag{2.20}$$

The precision and recall measures are both important, a classifier might have a good recall at the cost of a low precision, or viceversa, a way to combine these measures into one single measure of the effectiveness of the classifier is through the following equation:

$$F_\beta = \frac{(\beta^2 + 1) Precision \times Recall}{(\beta^2) Precision + Recall} \tag{2.21}$$

The β value might be seen as a weight for granting importance to either precision or recall, when $\beta < 1$ precision gains importance , but if $\beta > 1$ it is recall the favored value. When the value of β is chosen 1 the equation becomes the harmonic mean between precision and recall, this is called $F_1 - measure$.

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.22}$$

2.5 Semi-supervised learning for classification

On this work we tackled the problem of text classification without having labeled data from the domain, instead we have only labeled data from a different but related domain. In this case when we totally lack of labeled data, and also because the difficulty of getting enough amount of labeled data to build a fair supervised classifier, exploiting the information we can get from unlabeled data together with the available labeled data becomes a feasible option. We call this approach semi-supervised learning.

Semi-supervised learning might be transductive or inductive. We address transductive semi-supervised learning, when the solution is based on the correct inference of the labels for the unlabeled data; it only works over

the labeled and unlabeled training data, and cannot handle unseen data [47] . Inductive semi-supervised learning intends to inference a function for mapping the labels for the unlabeled data. Inductive learners are able to naturally handle unseen data.

The semi-supervised classification methods can be distinguished by those based on the classifier and those based on the data. Basically, classifier-based methods start by building a classifier with the available labeled data, so by an iterative process new training data is generated from the unlabeled data, then a new and more accurate model is built. The data-based methods try to discover the distribution of the labeled and unlabeled data together. Afterwards with the help of some assumptions it finds a function that describes the distribution in the best way. In this work we opted for a semi-supervised approach based on the classifier for adapting a classifier trained in a certain domain to perform in a different domain through an iterative process. Some semi-supervised methods are described next.

2.5.1 Generative Methods

Assuming a model $p(x, y) = p(y)p(x|y)$ where $p(x|y)$ is a mixture of distribution models, the mixture components can be identified with the help of a big amount of unlabeled data; then only few labelled data per component are needed in order to determine the mixture distribution.

2.5.2 Graph-based semi-supervised learning

A graph is constructed by making all the instances labeled and unlabeled its nodes. The nodes are joined by edges, the edges are weighted by the similarities between the instances. The objective is to find models that perform well on training data. Then a regularization is made, so that similar unlabeled nodes have similar labels. Graph methods are nonparametric, discriminative, and transductive in nature.

2.5.3 Co-training

In 1998 Mitchell and Blum proposed co-training [6]. The set of attributes describing the data is split in two disjoint subsets A and B, all the labeled examples are represented in terms of each of the two attribute subsets. Two

classifiers are built, each one trained with the data described in a different way. Classifier CA is trained with the data described with the A subset of attributes, and classifier CB is trained with the data described with the B subset of attributes. The most trustful labeled examples by the CA classifier are included in the training set for the CB classifier, also the most trustful labeled examples by the CB classifier are included in the training set for the CA classifier. The process continues until a number of iterations or a stop condition is met. The training progresses since these examples are at least as informative as random examples. As long as these two assumptions are true, there are at least two subsets of attributes able to describe the data well enough to train a good classifier, the two attribute subsets are conditionally independent given the class label. At the end two classifiers are obtained.

2.5.4 Self-training

The basic idea on self training was firstly documented by Yarowsky in 1995 [43]. In the self training technique an initial classifier is trained from the available labeled data. The classifier labels the unlabeled data, a metric is then applied to decide which of predictions are the most trustful. Instances labeled with the best confidence are then added to the former labeled data to obtain a new training set. A new classifier is trained with the current labeled data, this process is repeated until a stop condition is met. Self-training is a practical recommended method for a situation when the existing supervised classifier is hard to modify. Many natural language processing tasks have been treated with self-training, including word sense disambiguation[22], spam detection [34], automatic translation [7].

Both self training and co-training need 3 parameters to be defined, the number of iterations, the number of predictions chosen at each iteration to be included in the training set, and the number of unlabeled data to classify at each iteration, it could be all or just a subset of them.

2.6 Final comments

In this chapter we stated the approach of our method as a semi-supervised learning method, self-training to be more specific. We also detailed the learning algorithms we will use as gears in the mechanics of our method. Moreover

we explained how we can represent the text data in order to perform supervised learning. The evaluation of the text classification was presented too, as we detailed in the experimental settings, we only report accuracy since our test classification problems are balanced.

The next Chapter reviews some of the related works on the domain adaptation for text classification problems. It begins by defining the problem itself and its taxonomy. Afterwards some domain adaptation methods for text classification are depicted, all of them grouped by their approaches.

Chapter 3

Related work

In this Chapter we will define the domain adaptation problem starting from the transfer learning problem from which domain adaptation is part of. Once the problem is clear, we will review some of the state of the art methods that have dealt with this problem for text classification. Furthermore we will drop our method into one of the categories presented. At the end of the chapter we will discuss the traits of our method in comparison against the mentioned related work.

3.1 Transfer Learning

As said in the previous chapter, supervised learning algorithms are the main and most successful tool to solve text classification problems. In order to perform, supervised learning algorithms make two strong assumptions: first, they assume that the training and test data are drawn from the very same distribution; second, they also assume that the training and test data come from the same attribute space. Whenever any of these assumptions do not hold, the accuracy of the model generated from the training data will drop.

In real life, obtaining a sufficient amount of labeled data of any domain to solve a classification task is complicated and expensive. Gathering large amounts of new data requires great effort in many specific problems. The manual labeling of the gathered data demands a lot of time, and also needs domain experts as specialized as the domain requires. Whenever a reasonable

amount of labeled data from a different but related domain exists, it is desirable to benefit from the knowledge we can get from those data, even though they were meant to solve another task. Transfer learning has the purpose to intelligently apply knowledge learned previously to solve new problems[33] .

In 2010 Pan and Yang [33] formally define transfer learning as:

Given an auxiliary domain D_a and a learning task T_a, a target domain D_t and a learning task T_t, transfer learning aims to improve the learning of the target predictive function $f_t(\cdot)$ in D_t using the knowledge in D_a and T_a, where $D_a \neq D_t$, or $T_a \neq T_t$.

In the previous definition a domain D is composed of an attribute space V that describes the domain instances $X = v_1, v_2, ...v_n \in V$, and a marginal probability distribution $P(X)$. A learning task T for a domain $D = V, P(X)$ contains a label space Y and an objective predictive function $f(\cdot)$; the task is different from another if any of its elements is different. Two domains are considered related when they share a part, or there is a relation between their attribute spaces.

3.1.1 Inductive and Transductive transfer learning

There are two types of transfer learning. We call inductive transfer learning when we have labeled data available in the target domain, but the target task is different from the auxiliary task. We can benefit from the existing data to induce a classifier for the new task with some labeled examples from the target domain. Labeled data from an auxiliary domain are not required.

Transductive transfer learning occurs when we only have available labeled data from an auxiliary domain, but the target task is the same as the auxiliary task. This setting may appear from the following scenarios: the case when the marginal probability distributions of the two domains are different; and the case when both the attribute spaces and marginal probability distributions of the two domains are different. Transductive transfer learning makes use of the test data at the training time, the model should be learned with the unlabeled and the labeled data together.

In the transductive transfer learning case, when the marginal probability distributions of auxiliary and target data are different, we can adapt the predictive function obtained by the auxiliary data to perform on the same task on the target domain. The following section describes this kind of transductive transfer learning known as domain adaptation, which is the main subject treated on this work.

3.2 Domain Adaptation

When the source of knowledge to build a classifier, to accomplish a target task, is only the labeled data from an auxiliary domain D_a, the marginal probability distributions of the training set and the test set will be different[12]. The foregoing situation addresses the domain adaptation problem. The domain adaptation algorithms are designed to narrow the gap between the distributions of D_a and D_t[30].

"Domain adaptation is different from Semi-supervised learning. Semi-supervised learning methods employ both labeled and unlabeled data for better classification, in which the labeled and unlabeled data are assumed to come from the same domain"[14].

Domain adaptation is a real and current problem in natural language processing. On the next subsections we will review some of the state of the art methods for domain adaptation in this field. We have categorized the methods into four approaches. The first two are related on how data is represented and how the model is built; while the other two are related on how we can take advantage of the labeled and unlabeled data to approximate the distribution learned to the target domain. The last approach mentioned is important because this work is based on it.

3.2.1 Domain adaptation through common feature space discovery

The methods based on attribute space transformation look for a set of attributes with similar conditional distributions in the auxiliary domain D_a and the target domain D_t, and by projecting the data with this set of attributes, the difference between $P_a(y_i|x_j)$ and $P_t(y_i|x_j)$ is reduced.

In [5] a common attribute space meaningful in domains D_a and D_t is learned by a method called Structural Correspondence Learning (SCL). A set of pivot features are identified on the unlabeled data from both domains. Pivot features are elements occurring frequently in both domains which hold a similar behavior with the words that occur together with them. A binary classifier is trained for each pivot feature to determine the correlation between the non-pivot features to each pivot. All pivots are represented as a vector of weights that stands for the correlation of the pivot with non-pivot features. The main predictors are the top k singular vectors of the matrix which columns are the pivot weighted vectors. The main predictors are used to project each example vector, in order to get k new features to extend the current representation of the instance. Finally, a classifier is now trained with the new attribute space. If the pivots were chosen correctly, the common attribute representation will perform in both domains. The method was tested in the part of speech tagging task with a data corpus of the Wall Street journal as a source domain and a Medline biomedical texts for target domain.

An improvement for the SCL algorithm, SCL-MI was presented in [4]. The pivot selection is done through the computing of mutual information of the attributes to the labels in the auxiliary domain. The SCL-MI algorithm was tested in the sentiment analysis task, on a corpus of negative and positive opinions about products, books, dvds, electronics and kitchen appliances. Each of them were used as an auxiliary domain to learn the difference of positive and negative opinions and tested on each other domain; 12 combinations of domains were generated. SCL-MI outperformed the supervised baseline and its predecessor the SCL algorithm.

The correspondence matrix obtained by the pivot features correlations with other features in SCL, mixes up all features across domains. According to Ji et al.[21], this single view of the correspondence may lead to discover incorrect correlations, because there are words or expressions with completely different meaning when they appear in different domains. e.g. "read the book" in the opinions about movies domain could mean that a movie is boring or bad, and it is better to read the book in which the movie is based on. In contrast, the same sentence "read the book" in the opinions about books domain, recommends the book since it is good. To solve this problem [21]

proposes the method Multi-View Principal Component Analysis (MVPCA) for analyzing the correspondence between the features in every domain separately, they called this in a multi-view. First a set of bridge features are selected. These are the features in the intersection of the feature sets of both domains, with the highest mutual information between the feature and the class label. To extract the view of correspondence of a bridge feature X_{Λ_i} with the rest of the features, every instance in a domain is labeled as positive or negative according to the occurrence of X_{Λ_i} in each instance, whenever that feature is 0 in an instance the label is negative, positive otherwise. If the instance is positive the value of X_{Λ_i} is set as 0 in that instance so that feature does not have any predicting power, because all the instances will have 0. A linear classifier (logistic regression) is trained with all the pseudo-labeled instances in the domain, the separation hyperplane obtained represents the correspondence between the bridge feature and the rest of the features in that domain, this must be done for both domains, and every bridge feature. The set of correspondences in both domains make the multi-view. Principal component analysis (PCA) is applied. The obtained eigenvectors represent a low dimensional semantic space that minimizes the residual error. This is similar to the use of SVD in the SCL method. Finally the instances in both auxiliary and target domains are feature expanded with the matrix from the PCA step and a linear classifier is trained with the expanded auxiliary domain, the target domain is then classified. A Variant of MVPCA employs SCL before the PCA to add also the single view correspondences. The method was evaluated with the same sentiment analysis data corpus as SCL and 6 problems of the 20Newgroups corpus. Experiments were run for the cases were 100 and 200 bridge features were selected, also for different number of eigenvectors, 10, 30, 50 and 70. Results indicate that the algorithm performed better than SCL in 9 of 12 cases for the greater number of eigenvectors; the same happened for the 20Newgroups in 5 of 6 cases.

Daume III in [13] proposes a method for domain adaptation based on attribute space augmentation. This method was described by its author as frustratingly easy, and it was applied to the task of part of speech tagging. The idea consists of triplicate each attribute in different versions of itself, one specific for the auxiliary domain, one specific for the target domain and an independent. When the word takes the same use in both domains, the value in the independent copy will be high, if it has a specific most common use as

part of speech in the target domain it gets a high value in the target domain copy, the same stands for the auxiliary domain. Then a common supervised classifier is trained with the new attribute vectors. A generalized version might be described replicating of each attribute $K+1$ times, where K is the number of domains. Although it is very simple and easy to implement, the method is not easily applicable to other problems and areas, also it requires labeled data in the target domain as well as in the auxiliary domain.

One interesting approach within the attribute space transformation category is to use a new set of attributes representing groups of related attributes. Authors in [44] present Reinforcement Iterative Transferring Classification (RITC). It stands out for the interest of this work because it is also an iterative process and also addresses the cross-domain classification problem. The basic idea on this method is to exploit local information about the interrelationships between documents and words. This can improve the performance of classification over using global information, which changes from one domain to another. An iterative process is run, attribute vectors are calculated for documents and words, they represent the relation between words and documents. Each attribute takes the value of term frequency (tf) of the word in the document. Documents are labeled using a trained classifier in D_a with the documents represented in the recently calculated attribute space. The attribute vectors of words and documents are updated by clustering the words and documents, based on two user provided parameters, the number of word clusters and the number of document clusters. New features are based on the occurrence of words in clusters of documents. The experiments were made over three different cross-domain corpus, 20 Newsgroups, SRAA, and Reuters-21578. It overcame the supervised approach, represented by Naive Bayes and SVM. It also overcame in most of the cases transductive support vector machines, TSVM. The method showed to be able to perform similar with smaller training sets. Despite their good results reported, the method has two identifiable disadvantages. First an specific stop criterion is not suggested, they propose 5 iterations, determined in an empirical way. Second the method needs two parameters, they are chosen by the user, since there is not a specific way to determine good values.

3.2.2 Model adjustment based domain adaptation

The methods in this category aim directly to the model construction. Prior probabilities of the target domain might be very different from the prior probabilities of the auxiliary domain. A model built from the prior probabilities of the auxiliary domain will not fit to predict correctly in the target domain. It is necessary to re-estimate the priors to build a more accurate model for the target domain. Parameter estimation techniques are recurrent tools in machine learning, they help us to estimate the parameters of a model of the reality by taking advantage of the available data; but in the domain adaptation task, the distribution of target domain, and the distribution of auxiliary domain differs. The parameter estimation for domain adaptation should look for the criteria that promotes agreement between source and target expectations.

A model trained with labeled data will learn an approximation of the distribution of those data, so when tested on a different domain data it will not perform well, due to the different distributions. Some models might be adapted to improve the performance in a new distribution. Naive Bayes Transfer Classifier (NBTC)[11] estimates a classification model training Naive Bayes algorithm with the labeled domain data. Later the model is moved gradually to the target distribution with Expectation Maximization defined under the target's domain distribution. A local optimum of the maximum a posteriori hypothesis is found after a certain number of iterations. The trade-off parameters are estimated by the means of empirically fitting a function based on the Kullback-Leibler divergence. The method was tested on the SRAA, 20Newgroups, and Reuters-21578. It achieved a remarkable error rate reduction from using SVM and Naive Bayes algorithms trained with the labeled data of the auxiliary domain.

Although, the main idea of Kadar e Iria in TransferLF [25] is the labeling of features instead of documents, the main gear in the mechanism of the method is estimation of parameters. First a group of domain experts chose a set of words correlated with each class. Generalized expectation is used to find constraints of the model expectation for word-class combinations. An objective function is gotten combining two factors: first, the Kullback-Leibler divergence computed between the predicted labels distribution, on the set of instances with a certain attribute, and the reference distributions, estimated

with the help of the human expert feature labeling; second a regularization term, so the model does not have zeros in the parameters of unlabeled features. TransferLF estimates the parameters of a classification model by an optimization process of the objective function. TransferLDALF is a method is derived from TransferLF, it uses Latent Dirichlet Allocation (LDA)[2] for rising the number of labeled features with the attributes found in the target domain documents.

Both methods were tested in 2 datasets, the 20Newsgroups and SRAA corpus, and evaluated in accuracy. Both methods performed better than TSVM in 5 of the 9 problems evaluated, yet not the same cases. On contrast to requiring labeled instances, the method requires labeled features, which perhaps are easier to obtain, but still need manual labeling by an expert. In the presented experiments they used an oracle labeler instead a human expert. It employs mutual information over the label of true documents and features found on them. They define a threshold to select the labeled features, which lead us to the question of how many labeled features are necessary for the method to achieve good results, they empirically determine a threshold of 18 features per topic to get the best accuracy.

3.2.3 Domain Adaptation through instance selection and weighting

If we are able to identify some instances x_i in the auxiliary domain, for which $P_t(y_j|x_i) \neq P_a(y_j|x_i)$ is totally meet, those instances are noisy for approximating the distribution of the target domain. They might be removed from the training set to get a more accurate model. Either way if there exist in the target domain some instances for which $P_t(y_j|x_i) \simeq P_a(y_j|x_i)$ stands, if we manage to predict the labels for those instances, they might be used to make the training set more representative distribution of the target domain. Some learning algorithms allow to add a certain degree of importance to the training instances by weighting them. Based on the previous a weighting scheme is also plausible in order to balance the empirical distribution.

A way of taking advantage of instance and attribute weighting is by adjusting the weights iteratively by means of the optimization of a target function as in the bi-weighting method for domain adaptation [41]. The more useful

attributes for the classification are chosen, and the distributions of D_a and D_t got closer to each other. The method was tested in 6 datasets of web pages translated from Chinese to English, the Naive bayes and SVM classifiers were the chosen classifiers. The method got an $F_1 - measure$ in average of 0.79 with an averaged precision of 0.833 and an averaged recall of 0.754 the improvement was between 0.04 and 0.06, but it performed better than TSVM, TCA, information bottleneck and EDC.

In Pseudo in-domain data selection [1] the results of having very few data for training a statistical machine translation (SMT) outperforms the one trained with the complete corpus auxiliary domain. A set of the most relevant sentences to the target domain are drawn from the auxiliary domain, three criteria are proposed to evaluate the sentences to be chosen. By means of cross-entropy $H(s, LM_t)$ evaluating the perplexity $2^H(s, LM_t)$ of a language model of the target domain LM_t with the sentences s, the lowest perplexity sentences are chosen. The second criterion is evaluating the difference of the cross entropy between the sentences and the LM_t and LM_a, $H(s, LM_t) - H(s, LM_a)$. The last criterion called cross-entropy difference $[H_t(s, LM_a) - H_a(s, LM_a)] + [H_t(s, LM_t) - H_a(s, LM_t)]$. In the end a classifier algorithm is trained with the chosen sentences. This method has the advantage of the independence of the classifier, but it lacks of a threshold of selection.

3.2.4 Semi-supervised domain adaptation

An interesting semi-supervised algorithm called CDC for cross-domain text classification task is presented in [45]. The primary goal of the algorithm is to find the most useful instances from the labeled training set to classify the documents from the target domain. The hypothesis behind this algorithm is that these instances can be obtained through SVM, and that they are in fact the support vectors since they are the nearest points to the decision boundary. This method requires a small amount of labeled data from the target domain.

The CDC algorithm basically consists of the following steps: first a regular SVM classifier is trained with the labeled data from the auxiliary domain. The support vectors are taken as unlabeled data. The labels of the support vectors are predicted by performing semi-supervised classification (TSVM),

trained originally with the training labeled data from the target domain. The support vectors correctly classified are selected and with them and the training labeled data from the target domain train a new SVM classifier, finally this model is used to classify the unlabeled target domain data. Experiments with six cross-domain scenarios of the 20 Newsgroups corpus presented an error rate reduction, other experiment was done reducing the amount of labeled data available in the target domain.

Another semi-supervised method for transfer learning, based on self-training, is the one from [46] , they state the target domain and auxiliary domain share some common knowledge in the form of instances , they call those Common Knowledge Transmitters or CKT. A SVM classifier is induced with the labeled data of the auxiliary domain, the λ% classified instances labeled with the best confidence are chosen, λ% is a threshold defined as a percentage of the number of instances in the test set, the chosen CKT now become the training set, then a self-training process starts, using SVM as classifier, each iteration λ% labeled instances from the test set are inserted in the training test, this is done until a certain number of iterations is reached, or a stop criterion is met.

The experiments were run with the electronic version of Chinese Encyclopedia as the auxiliary domain, and the Chinese web documents collection as the target domain, both of them have the same set of classes; features were selected by chi-squared. The method improved the Micro-averaged F1-Measure by 8.92% to the standard supervised learning. There is not a defined way to choose the best λ, they tested different values of λ, from 5 to 100, stepped by 5, they saw that on the smallest values of λ the performance was not very good due to the lack of information in the training set, on the highest values the performance decreased because too much noise (mislabeled instances) was added to the training set. The remaining question is how to evaluate the confidence of the label predictions to decide which are the ones within the λ threshold.

3.3 Discussion

According to the definitions at the beginning of this chapter, auxiliary and target domains might differ from each other in their attributes or in their

marginal distributions. Methods in domain adaptation will make their way through these differences. Common feature space discovery methods rely on the attributes. Instance weighting and selection intends to get the marginal distribution closer during the training or before it. Model adjustment also goes for the distribution, but methods in this category can work on a distribution already learned, the distribution after training as well as during the training. Most of the methods in these three categories are somehow inductive semi-supervised since they take advantage of unlabeled data along to labeled data to induce a classifier, but the semi-supervised category is meant to contain those methods whose main idea is using a semi-supervised technique, either inductive or transductive.

We found the approach in [44] very interesting because it is iterative and it leads the training distribution to homogeneity in a subtle way. It changes the relations of the features to the documents on every iteration, which is important in text classification problems. We also believe that the representation of data is very powerful because it considers the two parts of the domain, attributes and marginal distribution. Despite the idea in [1] seems only applicable to the task it was designed for, statistical machine translation, we believe it can be taken for other natural language processing tasks.

Method	Requires target domain labeled data	Requires additional information	Depends on classifier	Useful for different tasks	User defined parameters
SCL[5]	NO	YES	NO	YES	NO
SCL-MI[4]	NO	YES	NO	YES	NO
MVPCA[21]	NO	NO	NO	YES	NO
FEDA[13]	YES	NO	NO	NO	NO
RITC[44]	NO	NO	NO	YES	3
NBTC[11]	NO	NO	YES	YES	1
TransferLF[25]	NO	YES	NO	YES	1
Bi-W[41]	NO	NO	NO	YES	2
Pseudo[1]	NO	NO	NO	NO	1
CDC[45]	YES	NO	YES	YES	NO
CKT[46]	NO	NO	YES	YES	2
Proposed Method	NO	NO	NO	YES	NO

Table 3.1: Comparison of the characteristics of the domain adaptation methods presented

All the previous methods proved to achieve good results in their respective testing tasks, yet there are some issues to comment about. Table 3.1 compares the characteristics of the presented methods, the last row of the table presents the method proposed in this work. The second column shows that in spite of them being domain adaptation methods, some still need a few labeled data from the target domain D_t. This is a disadvantage because these methods still need to answer how many labeled data from the target domain are required. [45] reports experiments using 20 labeled instances from D_t. It also reports the search of this number by reducing in 1% the number of training instances from D_a, starting from 10% in one of the six sets of the 20Newsgroups dataset. In the fourth column we can see three methods that require extra information resources to perform, in addition to labeled instances from an auxiliary domain. For [25] it is labeled features, they argue that labeling features is easier than labeling instances. Nevertheless, gathering data takes time, and the aid of human experts is required. The SCL methods [5][4] need a set of pivot features identified before the

application of the method, but these can be identified with an automatic procedure. Only two of the methods are specific to their respective task and cannot be used to solve a different one; the method in [1] was designed for part of speech tagging, and the method in [13] was designed for statistical machine translation.

The 6th column shows how many parameters defined by the user are needed. This seems to be a weakness of the current domain adaptation methods for text classification, consequently the solution to this problem still feels artisanal. All the iterative methods need a stop criterion, which might be just a certain number of iterations. Methods with a step of instance selection normally need a selection threshold as in [1] and [46]. Some other methods require special purpose parameters like [41] where two smooth factor parameters are needed, one for features weighting, and the other for instance weighting. The method in [25] needs to select a number of manually correlated features per topic. The method in [44] needs two user defined parameters, they present a parameter tunning section where they only show this tunning for one case, the performance of the method seems to depend very much on this two values, because the error rate is approximately 2 times greater in the worst case compared to the best case. The method in [46] needs also a very important parameter, the percentage of selected instances on every iteration. They clarify that the performance measured in accuracy and f1-measure depends on a good choice of this parameter. This last work is very related to our proposed method, since they are both based on using self-training for the domain adaptation.

In the next Chapter our proposed method is presented. We will expose the relation with previous works and we will explain the considerations about the chosen approach. Finally we will deepen into the method's working details.

Chapter 4

Proposed Method

In this chapter we will begin overviewing the forethought that lead us to the proposed method. We pass then to explain in detail the proposed method and its features. An alternative trend for the method is also presented. We end this chapter with the definition of our proposed stop criterion.

4.1 Introduction to the method

In the text classification problem we want to classify a set of unlabeled text documents from a certain domain. If it happens that there is not any labeled data from that target domain D_t to train a classifier, and instead we only have a labeled set of documents from a related auxiliary domain D_a, it becomes a domain adaptation problem. As the domains are related we could just train a classifier with the labeled data from D_a to help us to classify the set of text documents from D_t. Because of the difference of the underlying distributions and the feature spaces between D_t and D_a the performance of this classifier will be low. We can improve this situation by exploiting the knowledge from the D_t unlabeled data together with the labeled data from D_a. We propose to treat the domain adaptation for the text classification problem with a semi-supervised approach. Self-training seems almost naturally meant for bringing the D_a distribution close to the D_t distribution, because newly labeled data from D_t is added on each iteration. The new training set will have a more heterogeneous distribution.

Domain adaptation for text classification has been treated with self-training before. As we mentioned in the previous chapter the method in [46] has already used self-training for this task with interestingly good results. The method in [46] reported that it gained 8.92% in F1-measure compared to the standard supervised learning method SVM. We opted for this approach because discovering common knowledge and reinforcing the model with it is very attractive to address the problem of domain adaptation. The common knowledge should are considered the most confidently labeled instances from D_t. One advantage is that the domain adaptation is done gradually which assures confidence in the process. Another advantage of this approach is its simplicity in both the conceptual complexness method and the implementation.

A permanent question on self-training is what is the extent of the most confident labeled instances, how many instances we should choose on each iteration to enrich the training set. As mentioned in the previous chapter, one issue in the method of [46] is that its performance relies on the correct tuning of a parameter, the selection threshold. There is not a proposed way to determine a good value for the threshold. Measuring confidence of a prediction can be done by different means, but it is normally related to the mechanics of the classification algorithm employed. [46] does not mention how the confidence of predictions is estimated.

In this work we tried to go further in the domain adaptation for text classification problem treated with self-training. The scenario of domain adaptation with self-training will change on every iteration. The documents in D_a are the original training set, the documents in D_t are the original test set. On the following iterations the training set and test set change, the training set grows and the test set is reduced. The distribution on the training set becomes more heterogeneous to both domains. The feature space on the training set will change as well as in the test set, consequently the shared elements between them also change. The shared features are very important because it is actually this information which describes and categorizes the instances. Taking into account all the previous considerations, we propose a self-adjusted training approach method, which is able to adapt to the new distributions obtained on a self-training process. The method integrates some strategies to adjust itself each iteration. These strategies include the

feature space consideration, the instance selection criterion and the threshold of selection. The method is explained in detail in the following sections. This work also proposes 4 stop-criteria for the method.

4.2 A domain adaptation method for text classification based on a self-adjusting training approach

A linear classifier is trained with the labeled examples of the auxiliary domain. All the instances in the target domain are then classified. Support Vector Machines (SVM) algorithm was chosen because it is a very reliable classifier for text classification problems, SVM is able to handle large feature spaces and highly sparse vectors [26]; many text classification problems are linearly separable [24]. subsection 2.3.1 in Chapter 2 provides a deeper explanation about SVM. A linear Kernel was chosen for the SVM. A subset of the most confidently labeled instances of the target domain is selected, this subset is added to the training set. A new classifier is built with this new training set. The subset of the non-selected instances is classified again with the new classifier. The process repeats over until the test set becomes empty or a stop criterion is met. The figure 4.1 and algorithm 1 describe this method. The following sections detail the strategies for instance selection, determining threshold of selection and feature space considerations.

Instance selection with prototypes

The idea behind the proposed method for selecting the instances to be moved to the training set is that the most similar instances from the test set, to the instances in training set, are easier to classify for the built model, because they have more features in common or their marginal probability is similar to the learned distribution.

Lets think about instance selection as if it were an information retrieval task, we want to retrieve the most relevant documents from a collection for a given query. In this analogy our collection of documents where we can retrieve is the test set, and the query we need to satisfy is the class. Each class is represented by the prototype vector of the class. For the retrieving

Algorithm 1: Proposed method algorithm

Data: Da is a set of labeled documents of the auxiliary domain, each instance in Da is a pair (x_k,y_k), $x \in X$, a data point, and $y \in Y$, Y is the label space of the problem; Dt a set of unlabeled documents of the target domain to be classified

Result: A labeled set of documents Dt

$training\_set \leftarrow Da$;
$test\_set \leftarrow Dt$;
$i = 0$;
while $test\_set \neq \emptyset$ *or* $|selection_i| > 0$ **do**
    $feature\_set_i \leftarrow$
    $vocabulary\_in\_training\_set \bigcap vocabulary\_in\_test\_set$;
    represent $training\_set$ and $test\_set$ as vectors in $feature\_set_i$;
    train SVM Classifier with $training\_set$;
    classify $test\_set$ with the SVM model;
    $selection_i \leftarrow$ Instance_ selection($training\_set$, $test\_set$, Y);
    $training\_set \leftarrow training\_set \cup selection_i$;
    $test\_set \leftarrow test\_set - selection_i$;
    $i \leftarrow i + 1$
end
return $(training\_set - Da) \cup test\_set$

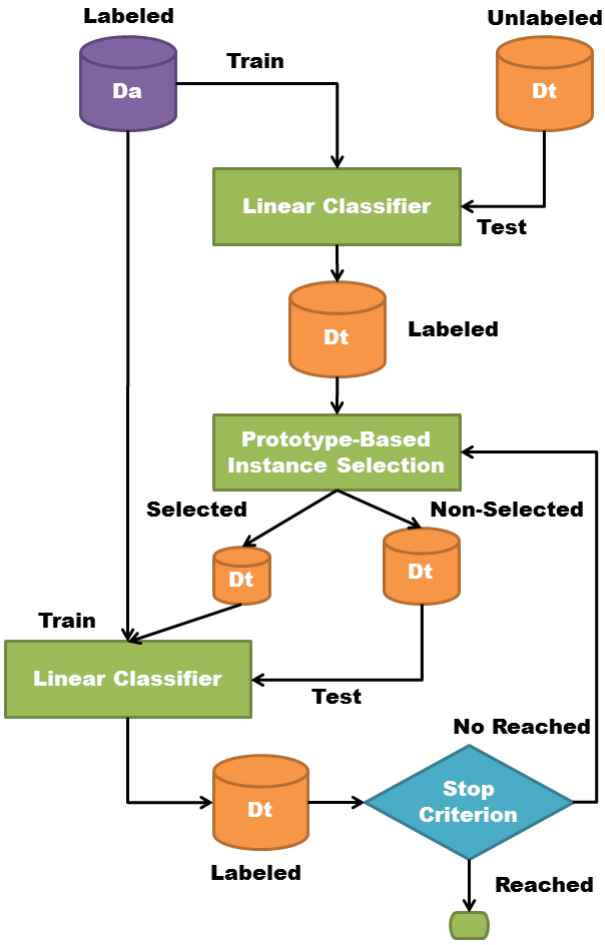

Figure 4.1: General diagram of the method

task we use a similarity measure. We compute the similarities between every instance in the test set with all the prototypes. Then we create a set of candidates to be selected, those are the instances which are nearer to the prototype of the class that SVM predicted than any other prototype.

$$argmax_{c \in C}(similarity(x_{t,i}, p_c)) \tag{4.1}$$

The purpose is to identify non-outlier instances, or the instances far from the soft margins that separate the classes of the model produced by SVM. The instances near the border between classes are meant to be more doubtfully

labeled. The instances near a prototype are more likely near a dense space, an instance surrounded by many examples of a certain class is very likely to be the same class. The prototype for each class was computed as the center of mass as explained on subsection 2.3.2. and equation 2.7 in Chapter 2.

The measure applied on this work, was cosine angular similarity, commonly useful on the information recovery task. The measure appears in Chapter 2, equation 2.10.

Selection threshold

As the distribution changes on each iteration for the training set, the center of mass of each class moves constantly. Having a fixed threshold in similarity or distance might have an unpredictable behavior in the amount of selected instances. In some iterations a prototype might be distant from most of the instances in the target domain selecting very few or even none of them. Contrarily a prototype might be very close to the target domain instances, therefore, a lot of them would be picked. If we have a fixed threshold as the number of instances to select each iteration, we fall again into the trouble of determining which is best value for the problem we are solving.

We propose a threshold that holds the idea of confidence, but it adapts on each iteration to the new distributions of the training set and test set. The threshold proposed is the mean (μ) plus a standard deviation(σ) of the similarity of the examples in the test set to the prototype of the class they were predicted. But we only consider the instances which are nearer to the prototype of the class that SVM predicted than any other prototype.

$$\mu = \frac{\sum_{k=1}^{L} distance(x_{c,k}, p_c)}{L} \tag{4.2}$$

$$\sigma = \sqrt{\frac{1}{L}\sum_{k=1}^{L}(x_k - \mu)^2} \tag{4.3}$$

$$threshold = \mu + \sigma \tag{4.4}$$

L is the number of instances that passed the first filter, i.e. all the instances closer to the prototype of the class they were predicted, $x_{c,k}$ is an instance predicted as the class c, p_c is the prototype of class c. The procedure 1 shows the pseudo-code for the instance selection method.

Procedure `Instance_ selection(`*training_set, test_set, C*`)`

preselect $= \emptyset$;
selection $= \emptyset$;
foreach *class* $c \in C$ **do**
    $p_c \leftarrow$Compute prototype of class y with the data $\in$ *training_set*;
    foreach $x \in$ *test_set* **do**
        $sim_{x,p_c} \leftarrow$similarity between x and p_c calculated with eq. 2.10
    end
end
foreach $x \in$ *test_set* **do**
    if *the label of* x *= the result of eq. 4.1* **then**
        preselect $\leftarrow$ *preselect* $\cup\, x$;
    end
end
$\mu \leftarrow$ mean of $sim_{x,p_{label\_of\_x}}$, $x \in$ *preselect* eq. 4.2;
$\sigma \leftarrow$ standard deviation of $sim_{x,p_{label\_of\_x}}$, $x \in$ *preselect* eq. 4.3;
threshold $= \mu + \sigma$;
foreach $x \in$ *preselect* **do**
    if $sim_{x,p_{label\_of\_x}} <$ *threshold* **then**
        selection $\leftarrow$ *selection* $\cup\, x$;
    end
end
return *selection*

4.2.1 Leaving the training wheels: The Auxiliary domain dropping

In order to approximate the predictive function to the target domain distribution, an alternative trend is set forth for the proposed method. We want the training set to be more representative of the D_t distribution, but the instance selection strategy previously shown uses the whole training set to

compute the prototypes. On the initial iteration we have no choice, since we only have labeled data from D_a. But at the beginning of the following iteration the training set starts to have some examples of the D_t distribution. However they get overwhelmed in the prototype computation, because the examples of D_a in the training set are still more. Hence the selection is still biased to the D_a distribution, and it will be for many iterations in the future. If we take only the labeled examples of D_t in the training set for the prototype construction, the domain adaptation should accelerate, also the accuracy of the classification should get better. The results exhibit the accuracy is better this way.

4.2.2 Feature space considerations

As defined on chapter 3, a domain D is composed of an attribute space V and a marginal distribution P(X), $D = \langle V, P(X) \rangle$. Two domains are related when they share part of these elements. Normally in text processing problems, the set of all different words found in the documents, the vocabulary, is taken as the attribute space. We could say the closeness of two domains of text depends on how many words in their vocabularies share, specially the non-stopwords. Two domains are more related when they have much vocabulary in common, nevertheless between two different domains the most of vocabulary is not shared. If we train our classifier in the feature space from the auxiliary domain, when we represent the target domain instances as vectors in that space all the columns of the non-shared attributes will become zeros. This does not necessarily mean that the attribute did not occur, yet It is very likely that the attribute does not even exist on that domain. We decided to take as a feature space the intersection of the auxiliary domain vocabulary and the target domain vocabulary. This is a set of attributes possibly meaningful in both domains. This results in shorter, and less sparse vectors.

In a self-training process, on each iteration some instances pass from the test set to enrich the training set and build a new model. When these new documents are added to the training set, new words are introduced to the training set, so its vocabulary grows. Also some of these new words are now shared with the documents left on the test set, growing the feature space. Later, due to the reduction of the test set some vocabulary might be lost on each iteration. When we take into consideration the previously

described situations, we realize that the feature space changes, and it must be recalculated every iteration.

From now on we will refer to the auxiliary domain vocabulary as Voc_{Da}, target domain vocabulary as Voc_{Dt}, and the feature space as $Voc_{Da \bigcap Dt}$.

4.2.3 Stop-criteria

One natural condition for the method to stop is when the job of classification is finished, this happens when the test set becomes empty because all the instances in D_t overcome the selection threshold. The second natural condition is when none instance overcomes the threshold, on this case the model trained in the next iteration would be the same as the current and no more changes of the model are possible. In both cases the process cannot continue. As only one is happening we call stop criterion CP4 when any of these situations happens.

As mentioned on last subsection when the new instances are introduced, new attributes are also introduced to the feature space. But this increment of attributes is not limitless. We observed that after some iterations many of the instances that hold the attributes in $Voc_{Da \bigcap Dt}$ have been moved to the training set, taking many of the attributes in the intersection with them, this attributes are now only present in the auxiliary domain. At this point the feature space will be reduced, and the model will lose information originally gotten from the target domain; we propose this point as a stop-criterion. Stopping when the size of the intersection of the vocabularies is smaller than the previous iteration, $\mid Voc_{Da \bigcap Dt_i} \mid < \mid Voc_{Da \bigcap Dt_{i-1}} \mid$. We call this stop criterion CP1.

The stop criterion CP2 occurs when the feature space size is smaller than the original at initial step, $\mid Voc_{Da \bigcap Dt_i} \mid < \mid Voc_{Da \bigcap Dt_0} \mid$.

We observed that as the iterations continue, the number of selected instances is reduced, this attends two reasons. First, the test set is smaller every time, and the instances still there are those filtered by the selection, so they are the hardest to be predicted. Second, when only very few instances

are selected, the model does not change considerably. Taking the aforementioned into consideration, we decided to use an arbitrary stop criterion for experimental purposes. The process stops when only one single instance from the test set is selected. From now on we will refer this as the stop criterion CP3.

4.3 Final comments

Now that the method has been explained along with the underlying ideas of its design. On the next Chapter we will define our experimental settings where the method was tested. Next Chapter also shows the results for testing scenarios, The analysis and discussion on the results will be found on Chapter 6.

Chapter 5

Experimental settings and results evaluation

In this chapter we delimit the experimental settings to evaluate the performance of the proposed method. At the very beginning we set the guidelines of evaluation. We also list the references to compare our results. Afterwards we explain some considerations of the experimental environment. We close this chapter by presenting our testing scenarios with their respective results.

5.1 Evaluating methodology

As we explained in the previous chapter, once an instance from the target domain is selected to become part of the training set, it maintains the last class they have been labeled with for the rest of the process. We have to keep in mind that what we are evaluating is the classification of the original test set, ergo the target domain documents. We need to gather all the instances from the target domain, those inside of the training set and those left in the test set until the last iteration. The latter ones with the last class label predicted to them. Then we are able to evaluate the complete target domain.

Experiments on the scenarios of cross-domain thematic text classification, and cross-domain sentiment analysis were done. These experiments are described in the following sections of this chapter.

Accuracy was the main metric for evaluating the methods. The accuracy gain from the traditional supervised approach applied for transfer learning is also presented. The Student's t-test and the Wilcoxon signed-rank test were used to evaluate the statistical significance of the results, this is discussed on the analysis of the results on the next Chapter.

5.2 Reference results

In this section we firstly define the reference values for analysis and discussion of the experiment results, the baselines and the upper bounds. We also describe how we obtained those reference values. Second we define the state of the art methods we chose for comparing the results obtained while testing the method on the datasets.

5.2.1 Baselines

As we wanted to measure the improvement in classification accuracy when our proposed method is applied, two of the best classifiers for the text classification task, Naive Bayes (NB) and Support Vector Machines (SVM), were chosen to acquire the reference results. The baseline represents the case where we use the auxiliary domain, related to the target domain, to serve as training set for building classifier for the target domain. We considered the accuracy obtained by such classifier as the worst we could go in the problem, also it is the starting point from where our method reduces the error in the classification on every iteration.

We took as the upper bound the case where we actually have plenty of labeled examples in the target domain. This reference result was calculated by implementing 10-fold cross-validation, using only the target domain documents with their respective labels, and averaging the accuracies. This would be the scenario when regular supervised learning is possible.

Tables 5.5, 5.6, 5.12 and 5.13 show the baseline, the accuracies of cross-domain supervised classification for each problem testing scenarios. The in-domain results are the average accuracy from the 10-fold cross-validation, only with the target domain instances. For comparing our method we always

take the Baseline obtained with SVM because its results were higher, and it corresponds to the first iteration of the method.

5.3 Global settings

In this section we list and describe the settings under which the experiments were run.

5.3.1 Preprocessing

Previously to an experiment all text documents were prepared with the following preprocessing. All the letters inside the texts were transformed to lowercase. Punctuation symbols and special characters were removed without exception. The words from a list of 174 stopwords in English [42] were also removed from the texts.

5.3.2 Data representation

The documents were represented as a bag of words, with a boolean weighting. For each document we obtained a vector with a length of the size of the vocabulary, filled with ones and zeros. We chose this kind of weighting over those based on the frequency in order to give all the importance to the presence of a shared terms from the target domain in the training set.

5.3.3 Other considerations

The implementation of the methods was made in python 2.7, with the *nltk* and *numpy* packages, and some bash shell scripts. The SVM classifier implementation used was libSVM 3.

5.4 Testing scenarios

In this section we present the experiments for testing the method in two domain adaptation tasks. The first task is cross-domain thematic categorization which corresponds to text classification. The second task is multi-sentiment analysis in a domain adaptation context. These tasks are defined in their own subsections. The experiments are presented by their task description

and objective, the dataset used, the results of the proposed method, and finally the results of the proposed method with auxiliary domain dropping.

5.4.1 Cross-domain thematic text categorization task

For the thematic categorization task, the objective is the same as the regular text categorization, described in the chapter 2. We want to assign a class label from a predefined set of classes to the documents in a set of unlabeled documents. The possible classes are subjects they are about. But in a cross-domain problem we will use an annotated corpus of a different domain as a training set for building a classifier, this auxiliary domain should be related to a certain extent to the target domain. The objective of this experiment is to test the performance of the proposed method for the task it was conceived.

Dataset

The dataset choice for the experiments on the cross-domain thematic text categorization task was the 20Newsgroups dataset[20], gathered by Ken Lang[27], and constituted by almost 20000 files. It receives its name because it is a collection of usenet articles from 20 different newsgroups. It is organized in a hierarchical form with seven mayor categories: comp, rec, sci, talk, misc, soc, alt, each of them encompassing several subcategories. The following table shows all the available subcategories grouped by related subjects.

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.guns talk.religion.misc	alt.atheism soc.religion.christian

Table 5.1: Categories related among the subcategories of 20Newsgroups

The comp category is about computing, rec stands for sports records, sci included sciences like cryptography, electronics, medicine, and outer space, talk is filled with talks about different subjects on politics and religion, alt and soc are about religion and misc includes varied topics. We built 6 different cross-domain problems recurrent in the literature[44][11][25] with the classes comp, rec, sci and talk. By taking advantage of the subcategories we can create different domains for the same task, for example a domain of rec might be formed with the documents about baseball and motorcycles, while some other domain of rec could incorporate documents about cars and hockey, despite of they are related, and they are both about sports records, the words, subjects and the way to talk about hockey is very different from baseball. The following table shows the structure of these 6 problems.

#	Problem	Target domain (Dt)	Auxiliary domain (Da)
1	comp_vs_rec	comp.os.ms-windows.misc comp.windows.x rec.autos rec.sport.baseball	comp.graphics comp.sys.ibm.pc.hardware rec.motorcycles rec.sport.hockey
2	comp_vs_sci	comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x sci.med sci.space	comp.graphics comp.os.ms-windows.misc sci.crypt sci.electronics
3	comp_vs_talk	comp.os.ms-windows.misc comp.sys.ibm.pc.hardware talk.politics.guns talk.politics.misc	comp.graphics comp.sys.mac.hardware comp.windows.x talk.politics.mideast talk.religion.misc
4	rec_vs_sci	rec.motorcycles rec.sport.hockey sci.crypt sci.electronics	rec.autos rec.sport.baseball sci.med sci.space
5	rec_vs_talk	rec.sport.baseball rec.sport.hockey talk.politics.mideast talk.religion.misc	rec.motorcycles rec.autos talk.politics.guns talk.politics.misc
6	sci_vs_talk	sci.crypt sci.space talk.politics.guns talk.politics.mideast	sci.electronics sci.med talk.politics,misc talk.religion.misc

Table 5.2: Domain adaptation data set problems 20Newgroups

The next tables provide information about each problem. It is worth to mention that duplicates of the documents were removed, all the headers with metadata on each document were also eliminated.

#	Problem	$\|Da\|$	$\|Dt\|$
1	comp_vs_rec	4909	3949
2	comp_vs_sci	3930	4900
3	comp_vs_talk	4482	3652
4	rec_vs_sci	3961	3965
5	rec_vs_talk	3669	3561
6	sci_vs_talk	3374	3828

Table 5.3: Number of documents in each domain per problem

#	Problem	$\|Voc_{Da}\|$	$\|Voc_{Dt}\|$	$\|Voc_{Da \cup Dt}\|$	$\|Voc_{Da \cap Dt}\|$	$\%\|Voc_{Da \cap Dt}\|$
1	comp_vs_rec	42881	62259	90081	15059	16.71%
2	comp_vs_sci	59265	53871	96524	16612	17.21%
3	comp_vs_talk	55098	57693	95603	17188	17.97%
4	rec_vs_sci	42163	38962	65337	15788	24.16%
5	rec_vs_talk	37182	41125	62488	15819	25.31%
6	sci_vs_talk	41355	45912	68866	18401	26.72%

Table 5.4: Features on each Domain,$\%|Voc_{Da \cap Dt}|$ is the percentage of shared by the two domains

Baseline and upper bound

#	Problem	Baseline (SVM)	in-domain (SVM)	relative diff.
1	comp_vs_rec	84.55	98.30	16.26%
2	comp_vs_sci	67	97.18	45.05%
3	comp_vs_talk	88.99	98.96	11.20%
4	rec_vs_sci	75.58	98.44	30.24%
5	rec_vs_talk	71.38	98.96	38.63%
6	sci_vs_talk	73.58	98.67	34.09%

Table 5.5: Baselines and upper bound for 20Newsgroups problems using SVM

#	Problem	Baseline (NB)	in-domain (NB)	relative diff.
1	comp_vs_rec	86.12	95.34	9.67%
2	comp_vs_sci	74.49	91.55	18.63%
3	comp_vs_talk	87.68	94.11	6.83%
4	rec_vs_sci	81.29	94.27	13.77%
5	rec_vs_talk	67.70	92.53	26.83%
6	sci_vs_talk	66.37	91.44	27.17%

Table 5.6: Baselines and upper bound for 20Newsgroups problems using Naive Bayes

As we may observe, there is a big difference between the accuracy obtained by training the classifiers with in-domain data, and the accuracy obtained by using an auxiliary domain, which is considerably lower. Relative differences go from 11.20% to 45.05% and an average of 29.24%. The high accuracies gotten from a training set with the same distribution as the test set, 98.41 in average for SVM, lead us to think that even a small fraction of instances from the target domain can strongly contribute to obtain a better model for the target domain.

Results of the proposed method

The table 5.7 shows the results of applying the proposed method for a cross-domain thematic text classification task, the 20Newsgroups dataset problems. The accuracy on the classification accomplished is shown in the columns CP1, CP2, CP3 and CP4, each one of them corresponds to a stop criterion, the stop criteria are described at the end of chapter 4. The i columns display the number of iterations needed to reach the stop criterion on their left. Starting off from the baseline, which is actually the result of first iteration, we may observe on every case an improvement on the accuracy from our baseline on every stop criterion reached.

#	Problem	Baseline	CP1	i	CP2	i	CP3	i	CP4	i
1	comp_vs_rec	84.55	88.85	3	90.52	5	91.21	40	**91.39**	65
2	comp_vs_sci	67.00	76.22	3	80.91	9	82.63	33	**82.95**	60
3	comp_vs_talk	88.99	92.14	2	93.15	3	**96.24**	27	96.13	65
4	rec_vs_sci	75.58	84.08	3	88.14	8	91.85	62	**91.90**	72
5	rec_vs_talk	71.38	81.07	3	83.40	8	85.65	37	**86.15**	61
6	sci_vs_talk	73.58	80.69	4	82.65	7	84.92	29	**90.70**	74

Table 5.7: Accuracies reached with our proposed method over 20Newsgroups per stop-criterion

The improvement in the accuracy through this method on the 20Newsgroups problems is presented on table 5.8. The third column holds the baseline accuracy obtained for each problem, CP columns hold the relative accuracy gain, that is the percentage of how much the accuracy improved when each stop criterion was reached. The columns marked with an i show the number of iterations needed for reaching the stop criterion at their left.

#	Problem	Baseline	CP1	i	CP2	i	CP3	i	CP4	i
1	comp_vs_rec	84.55	5.09	3	7.06	5	7.88	40	8.08	65
2	comp_vs_sci	67.00	13.76	3	20.76	9	23.33	33	23.80	60
3	comp_vs_talk	88.99	3.54	2	4.67	3	8.15	27	8.02	65
4	rec_vs_sci	75.58	11.24	3	16.61	8	21.53	62	21.59	72
5	rec_vs_talk	71.38	13.57	3	16.84	8	19.99	37	20.69	61
6	sci_vs_talk	73.58	9.66	4	12.33	7	15.41	29	23.27	74

Table 5.8: Percentage of accuracy gained by using the proposed method over 20Newsgroups per stop-criterion, All the values of CP columns are percentages

Results of the proposed method with auxiliary domain dropping

As we may observe, the accuracy achieved with the proposed method with auxiliary domain dropping is higher, except for the case of sci_vs_talk, where the proposed method without auxiliary domain dropping was 1.55 better than the proposed method with auxiliary domain dropping. Also 4 of the 6 cases reached the CP4 stop criterion in less iterations. Only in one case the

method without auxiliary domain dropping reached CP4 in a smaller number of iterations.

#	Problem	Baseline	CP1	i	CP2	i	CP3	i	C4	i
1	comp_vs_rec	84.55	87.99	2	90.48	4	93.82	39	**93.92**	49
2	comp_vs_sci	67.00	77.43	3	83.51	9	85.71	49	**86.71**	102
3	comp_vs_talk	88.99	90.61	2	93.04	3	**97.28**	32	97.26	37
4	rec_vs_sci	75.58	83.58	3	87.79	7	92.10	48	**92.15**	57
5	rec_vs_talk	71.38	82.56	3	85.36	7	90.87	59	**91.00**	61
6	sci_vs_talk	73.58	77.79	2	85.44	7	88.82	37	**89.15**	60

Table 5.9: Accuracies for using the proposed method with auxiliary domain dropping over 20Newsgroups per stop-criterion

#	Problem	Baseline	CP1	i	CP2	i	CP3	i	CP4	i
1	comp_vs_rec	84.55	4.06	3	7.01	5	10.96	40	11.08	65
2	comp_vs_sci	67.00	15.57	3	24.64	9	27.92	33	29.42	60
3	comp_vs_talk	88.99	1.82	2	4.55	3	9.31	27	9.29	65
4	rec_vs_sci	75.58	10.58	3	16.15	8	21.86	62	21.92	72
5	rec_vs_talk	71.38	15.66	3	19.58	8	27.30	37	27.49	61
6	sci_vs_talk	73.58	5.72	4	16.12	7	20.71	29	21.16	74

Table 5.10: Percentage of accuracy gained by using the proposed method with auxiliary domain dropping over 20Newsgroups per stop-criterion, All the values of CP columns are percentages

5.4.2 Cross-domain sentiment analysis task

In this task we pretend to identify the opinion or sentiment expressed by the author on a text document. In this work the experiments were run over the specific task of polarity detection where the objective is to identify when an opinion was positive or negative.

In the cross-domain scenario, we only have available a labeled set of opinion documents on one particular domain, and we need to classify documents expressing positive or negative opinions from a different domain. As in the

thematic scenario, the vocabulary used for describing an opinion about something might be very different from one domain to another. Lets take as example opinions about products, some features important in one domain simply do not exist on other. A fridge might be energy-saving but a pair of pants cannot; furthermore the expressions and words from a domain may have distinct or even opposite meaning from one domain to another. The phrase "read the book" in a book review should be positive, since it is very probable that it encourages the reader of the review to actually read the book because it is interesting. On the other hand, if we find "read the book" in a movie review, it could mean that the book is much better than the movie, and maybe they are suggesting to read it instead of watching the movie.

Dataset

For the sake of evaluating the method on the sentiment analysis scenario, the John Blitzer and Mark Dredze's Multi-Domain Sentiment Dataset (version 2.0) [3] was chosen. This dataset is a collection of customer reviews of amazon.com for several different categories of products. Each category might be seen as a domain. The reviews have a rate of 5 stars, this rate was mapped to binary class labels, reviews with more than 3 stars were considered positive while reviews with less than 3 were considered negative, reviews with 3 stars were discarded. Works as [4] [5] [21] also use this corpus.

Twelve binary problems were formed from all the permutations of the domains books, electronics, dvd, and kitchen. In contrast to the other scenario where the problems were named by the classes, these problems are named with the domains, first the auxiliary domain, followed by _vs_ and then the target domain. All domains are balanced, they have 2000 domain with 1000 documents per class.

#	Problem	$\|Voc_{Da}\|$	$\|Voc_{Dt}\|$	$\|Voc_{Da \cup Dt}\|$	$\|Voc_{Da \cap Dt}\|$	$\%\|Voc_{Da \cap Dt}\|$
1	electronics_vs_dvd	106356	182344	262411	26289	10.01%
2	kitchen_vs_dvd	90368	182344	249200	23512	9.43%
3	books_vs_electronics	190087	106356	269919	26524	9.82%
4	dvd_vs_books	182344	190087	326264	46167	14.15%
5	kitchen_vs_electronics	90368	106356	172635	24089	13.95%
6	electronics_vs_books	106356	190087	269919	26524	9.82%
7	books_vs_kitchen	190087	90368	256471	23984	9.35%
8	kitchen_vs_books	90368	190087	256471	23984	9.35%
9	dvd_vs_kitchen	182344	90368	249200	23512	9.43%
10	dvd_vs_electronics	182344	106356	262411	26289	10.01%
11	electronics_vs_kitchen	106356	90368	172635	24089	13.95%
12	books_vs_dvd	190087	182344	326264	46167	14.15%

Table 5.11: Features on each Domain,$\%|Voc_{Da \cap Dt}|$ is the percentage of shared by the two domains

On the sentiment analysis scenario the underscore character was not removed, because it was part of some attributes, since the version employed was already pre-processed.

Baseline and upper bound

#	Problem	Baseline (SVM)	in-domain (SVM)	relative diff.
1	electronics_vs_dvd	70.25	81.45	15.94%
2	kitchen_vs_dvd	73.45	81.45	10.89%
3	books_vs_electronics	74.4	85.3	14.65%
4	dvd_vs_books	76	81.75	7.56%
5	kitchen_vs_electronics	82.15	85.30	3.83%
6	electronics_vs_books	66.15	81.75	23.58%
7	books_vs_kitchen	77.30	88.85	14.94%
8	kitchen_vs_books	71.05	81.75	15.06%
9	dvd_vs_kitchen	75.05	88.85	18.39%
10	dvd_vs_electronics	74.8	85.30	14.04%
11	electronics_vs_kitchen	82.95	88.85	7.11%
12	books_vs_dvd	79.95	81.45	1.88%

Table 5.12: Baselines multi-sentiment database problems using SVM

#	Problem	Baseline (NB)	in-domain (NB)	relative diff.
1	electronics_vs_dvd	68.25	80.85	18.46%
2	kitchen_vs_dvd	71.50	80.85	13.08%
3	books_vs_electronics	70.10	82.05	17.05%
4	dvd_vs_books	74.55	77.55	4.02%
5	kitchen_vs_electronics	78.40	82.05	4.65%
6	electronics_vs_books	68.10	77.55	13.88%
7	books_vs_kitchen	71.35	83.15	16.54%
8	kitchen_vs_books	69.05	77.55	12.31%
9	dvd_vs_kitchen	74.75	83.15	11.24%
10	dvd_vs_electronics	72.40	82.05	13.33%
11	electronics_vs_kitchen	81.45	83.15	2.09%
12	books_vs_dvd	72.40	80.85	11.67%

Table 5.13: Baselines multi-sentiment database problems using Naive Bayes

For the Amazon dataset, classified with SVM, relative differences go from 1.87% to 23.58% and an average of 12.32%. Similar differences are hold while using Naive Bayes.

Results of the proposed method

Table 5.14 shows the results of applying the proposed solution method for a cross-domain sentiment analysis task, the Amazon dataset problems. On this scenario the method still raised the accuracy of classification on every problem, except for kitchen_vs_books, where 0.5% was lost against the baseline. Error reduction appears on table 5.15.

#	Problem	Baseline	CP1	i	CP2	i	CP3	i	C4	i
1	electronics_vs_dvd	70.25	72	3	72.85	8	74.2	28	**75.3**	60
2	kitchen_vs_dvd	73.45	72.35	3	73.6	10	74.65	30	**74.95**	70
3	books_vs_electronics	74.4	74.3	2	74.3	2	**79.55**	37	79.5	38
4	dvd_vs_books	76	76.1	1	76.1	1	76.65	35	**77.15**	76
5	kitchen_vs_electronics	82.15	82.25	2	82.05	3	**82.65**	40	82.59	59
6	electronics_vs_books	66.15	66.3	3	67.7	9	69.5	31	**70.55**	66
7	books_vs_kitchen	77.3	78.55	2	79.15	3	81.65	36	**81.9**	63
8	kitchen_vs_books	**71.05**	69.35	3	69.15	13	70.3	38	70.55	63
9	dvd_vs_kitchen	75.05	75.75	2	77.15	3	80.7	26	**81.55**	81
10	dvd_vs_electronics	74.8	76.2	2	76.3	3	80	32	**80.2**	47
11	electronics_vs_kitchen	82.95	83.7	2	83.1	3	83.8	27	**83.8**	54
12	books_vs_dvd	79.95	79.7	1	79.7	1	79.5	31	**80.15**	57

Table 5.14: Accuracies the proposed method over multi-sentiment database per stop-criterion

#	Problem	Baseline	CP1	i	CP2	i	CP3	i	C4	i
1	electronics_vs_dvd	70.25	2.49	3	3.70	8	5.62	28	7.19	60
2	kitchen_vs_dvd	73.45	-1.50	3	0.20	10	1.63	30	2.04	70
3	books_vs_electronics	74.40	-0.13	2	-0.13	2	6.92	37	6.85	38
4	dvd_vs_books	76.00	0.13	1	0.13	1	0.85	35	1.51	76
5	kitchen_vs_electronics	82.15	0.12	2	-0.12	3	0.61	40	0.53	59
6	electronics_vs_books	66.15	0.23	3	2.34	9	5.06	31	6.65	66
7	books_vs_kitchen	77.30	1.62	2	2.39	3	5.63	36	5.95	63
8	kitchen_vs_books	71.05	-2.39	3	-2.67	13	-1.05	38	-0.70	63
9	dvd_vs_kitchen	75.05	0.93	2	2.80	3	7.53	26	8.66	81
10	dvd_vs_electronics	74.8	1.87	2	2.00	3	6.95	32	7.21	47
11	electronics_vs_kitchen	82.95	0.90	2	0.18	3	1.02	27	1.02	54
12	books_vs_dvd	79.95	-0.31	1	-0.31	1	-0.56	31	0.25	57

Table 5.15: Percentage of accuracy gained by using the proposed method over multi-sentiment database per stop-criterion, all the values of CP columns are percentages

Results of the proposed method with auxiliary domain dropping

Table 5.16 shows that again the proposed method with auxiliary domain dropping produced a more accurate classification than the proposed method without auxiliary domain dropping, except for the books_vs_dvd case.

#	Problem	Baseline	CP1	i	CP2	i	CP3	i	C4	i
1	electronics_vs_dvd	70.25	72.75	3	73.45	9	75.60	31	**76.45**	70
2	kitchen_vs_dvd	73.45	73.30	3	75.50	10	**76.35**	33	76.30	54
3	books_vs_electronics	74.40	74.35	2	76.15	3	**79.80**	39	79.75	62
4	dvd_vs_books	76.00	76.10	1	76.10	1	**78.70**	38	**78.70**	59
5	kitchen_vs_electronics	82.15	82.40	2	82.45	3	**82.90**	49	82.70	62
6	electronics_vs_books	66.15	68.30	3	71.85	14	73.20	32	**73.80**	65
7	books_vs_kitchen	77.30	79.05	2	80.05	3	81.80	27	**82.30**	64
8	kitchen_vs_books	71.05	72.85	4	74.10	15	74.55	39	**74.80**	72
9	dvd_vs_kitchen	75.05	76.70	2	79.50	4	82.25	24	**82.55**	59
10	dvd_vs_electronics	74.80	76.55	2	77.00	3	79.90	29	**80.65**	50
11	electronics_vs_kitchen	82.95	83.50	2	83.15	3	84.46	37	**84.75**	66
12	books_vs_dvd	**79.95**	79.70	1	79.70	1	79.50	30	79.80	86

Table 5.16: Accuracies for the proposed method with auxiliary domain dropping over multi-sentiment database per stop-criterion

#	Problem	Baseline	CP1	i	CP2	i	CP3	i	CP4	i
1	electronics_vs_dvd	70.25	3.56	3	4.55	8	7.61	28	8.82	60
2	kitchen_vs_dvd	73.45	-0.20	3	2.79	10	3.95	30	3.88	70
3	books_vs_electronics	74.40	-0.06	2	2.35	2	7.26	37	7.19	38
4	dvd_vs_books	76.00	0.13	1	0.13	1	3.55	35	3.55	76
5	kitchen_vs_electronics	82.15	0.30	2	0.36	3	0.91	40	0.66	59
6	electronics_vs_books	66.15	3.25	3	8.61	9	10.66	31	11.56	66
7	books_vs_kitchen	77.30	2.26	2	3.56	3	5.82	36	6.47	63
8	kitchen_vs_books	71.05	2.53	4	4.29	13	4.92	38	5.28	63
9	dvd_vs_kitchen	75.05	2.20	2	5.92	3	9.59	26	9.99	81
10	dvd_vs_electronics	74.80	2.34	2	2.94	3	6.82	32	7.82	47
11	electronics_vs_kitchen	82.95	0.66	2	0.24	3	1.82	27	2.17	54
12	books_vs_dvd	79.95	-0.31	1	-0.31	1	-0.56	31	-0.19	57

Table 5.17: Percentage of accuracy gained by using the proposed method with auxiliary domain dropping over multi-sentiment database per stop-criterion, All the values of CP columns are percentages

5.5 Final comments

According to the results, the method is effective for thematic cross-domain text classification, achieving accuracies from 82.95% to 96.13%, 89.87% on average. It also proved competent for reducing the error considerably against the supervised classification on the thematic task, up to a 66.83%. The accuracy was improved in all the cases.

For the cross-domain sentiment analysis scenario the results were also good, the accuracy was improved in 17 of the 18 cases. The method reduced the error in this case from 0.99% to 26.05%, 11.23% in average. The case where we obtained a lower performance than baseline was only a difference of 1.72%.

The analysis of the results is left for the next chapter. We will discuss the observations, and put them on test to assure significance. A comparison with other methods is also found on the next chapter.

Chapter 6

Analysis and Discussion

In this Chapter we analyze the results of the experiments. The comparative between dropping auxiliary domains is also presented in this Chapter. We present the results of the comparison of the method with other work on domain adaptations for text classification. We also discuss the features of our proposed method. Finally we try to measure the gap between the distributions of the domains in order to know the difficulty of the problems from our data, in addition to look for a correlation between the problem difficulty and the effectiveness of the proposed method on each problem.

6.1 Self-adjusted training

For checking the effectiveness of the proposed method in improving the accuracy of classification, we used a paired-samples t-test with the results of CP4 for all the 18 problems and the baselines. The null hypothesis is that no change in the accuracy of classification was achieved by using the method, than the accuracy gotten from just training a supervised classifier with an auxiliary domain. The test rejected the null hypothesis, showing the proposed method to be associated with a statistically significant better accuracy than the one obtained by training a supervised classifier with an auxiliary domain, $t(17) = -4.5115, p = 0.0003, \alpha = 0.05$ (proposed method $\mu = 82.08$; baseline $\mu = 75.81$).

6.2 Self-adjusted training vs Self-adjusted training with auxiliary domain dropping

If we compare the number of iterations taken on average by the proposed method without and with auxiliary domain dropping to reach the stop criteria, it seems that there is not too much difference. In order to evaluate if the auxiliary domain dropping actually improved the results obtained by the proposed method, we used a paired-samples t-test, and a Wilcoxon signed-ranks test, taking the accuracy of the classification reached at CP4 in the 18 problems studied, with auxiliary domain dropping and without auxiliary domain dropping. The null hypothesis was that no change in the accuracy of classification was achieved by using the auxiliary domain dropping in the proposed method. Both paired-samples t-test $[t(17) = -3.53, p = 0.003, \alpha = 0.05]$ and the Wilcoxon signed-ranks test $[z = -3.0057, p = 0.0026, w = 16.5]$ indicated that when the proposed method is implemented with auxiliary domain dropping, the accuracy it achieves is significantly higher than method without using auxiliary domain dropping. (without auxiliary domain dropping $\mu = 82.08, median = 81.72$; with auxiliary domain dropping $\mu = 83.5, median = 82.42$).

6.3 Comparison with other methods

This section presents the results of comparing the proposed method with other state of the art methods. The metric compared was accuracy. The selection of the state of the art works to compare the results of our method, was related with the available datasets chosen for the testing scenarios. The accuracy from the state of art methods is the one reported in their respective publications, as they used the same datasets for testing. In the case of cross-domain thematic text classification we compared our results with the ones obtained with the following works: [21],[45], and [5]. For the cross-domain sentiment analysis we compared our results with [5] and [4]. Chapter 3 includes a description of these methods.

Despite [46] is the most related work to our method, a comparison with it was not possible. They use a very specific dataset which was not available to us. Due to the lack of details on how they select instances implementing the method was not an option either.

Table 6.1 presents the comparison on the 20Newsgroups database for the cross-domain thematic text classification. The column MVPCA corresponds to [21], the column SCL corresponds to [5] and CDC corresponds to [45]. The last two columns show the results achieved by the method proposed in this work, without and with auxiliary domain dropping respectively. The method with auxiliary domain dropping got the best results in 3 out of 6 cases. The method without auxiliary domain dropping, SCL and CDC overcome the other in one case each. The MVPCA and SCL require a parameter m, how many labeled features are used as bridge features. The value of the parameter was 100. They also need another parameter p, which is the number of uniform orthogonal vectors, this value was 50. The proposed method accuracies are for CP4, because as we said on Chapter 4, it is its natural finish.

#	Problem	MVPCA	SCL	CDC	Self-adjusted training	Self-adjusted training D_a dropping
1	comp_vs_rec	88.17	65.86	93.37	91.39	**93.92**
2	comp_vs_sci	71.81	**88.55**	80.67	82.95	86.71
3	comp_vs_talk	93.68	91.2	96.79	96.13	**97.26**
4	rec_vs_sci	79.38	76.31	**92.9**	91.9	92.15
5	rec_vs_talk	76.01	62.28	72.26	86.15	**91**
6	sci_vs_talk	75.49	75.1	83.12	**90.7**	89.15

Table 6.1: Accuracy comparison with methods MVPCA, SCL and CDC with the 20Newsgroups Dataset

#		SCL	SCL-MI	Self-adjusted training	Self-adjusted training D_a dropping
1	electronics_vs_dvd	74.3	76.2	75.3	**76.45**
2	kitchen_vs_dvd	75.4	**76.9**	74.95	76.3
3	books_vs_electronics	77.5	75.9	79.5	**79.75**
4	dvd_vs_books	76.8	**79.7**	77.15	78.7
5	kitchen_vs_electronics	83.7	**86.8**	82.59	82.7
6	electronics_vs_books	**75.4**	**75.4**	70.55	73.8
7	books_vs_kitchen	78.7	78.9	81.9	**82.3**
8	kitchen_vs_books	66.1	68.6	70.55	**74.8**
9	dvd_vs_kitchen	79.4	81.4	81.55	**82.55**
10	dvd_vs_electronics	74.1	74.1	80.2	**80.65**
11	electronics_vs_kitchen	84.4	**85.9**	83.8	84.75
12	books_vs_dvd	74	75.8	**80.15**	79.8

Table 6.2: Accuracy comparison with methods SCL and SCL-MI with the Amazon Dataset

The table 6.2 shows the comparison on the Amazon database for the cross-domain sentiment analysis. The method was compared with SCL [5] and SCL-MI [4]. The proposed method with auxiliary domain dropping performed better on 6 of 12 cases. SCL-MI beated the other method in 5 cases. The last case was won by the proposed method without using auxiliary domain dropping, nevertheless the proposed method with auxiliary domain dropping performed better on this case than both SCL and SCL MI.

6.4 Stop criteria analysis

In the natural stop criterion CP4, the accuracy in the classification obtained was the best in comparison with the other stop criteria in 15 of the 18 problems for the proposed method, and 14 of the 18 for the proposed method using auxiliary domain dropping.

	CP1	CP2	CP3	CP4
Proposed method average iterations	2.44	5.50	34.38	62.83
Proposed method with auxiliary domain dropping average iterations	2.33	5.88	37.33	63.55

Table 6.3: Average of iterations per stop criterion

We evaluated how suitable are the stop criteria, for obtaining the best accuracy in the less number of iterations. Obviously each stop criterion would take more iterations or at least the same as the last one. So we need to compare if the accuracy gaining from a stop criterion to the previous is significant, we did this by making a paired-samples t-test for each criterion and its following, CP1 with CP2, CP2 with CP3, and CP3 with CP4. When the accuracy gaining at certain stop criterion is not significant against the previous stop criterion, then previous stop criterion is better, since it reached similar results in less number of iterations.

Stop criteria	t(17)	p	α
CP1 ($\mu = 78.31$), CP2 ($\mu = 79.44$)	-3.30	0.0042	0.05
CP2 ($\mu = 79.44$), CP3 ($\mu = 81.42$)	-5.88	0.000018	0.05
CP3 ($\mu = 81.42$), CP4 ($\mu = 82.07$)	-2.08	0.0526	0.05

Table 6.4: paired-sample ttests for stop criteria in method 1

Stop criteria	t(17)	p	α
CP1 ($\mu = 78.69$), CP2 ($\mu = 80.81$)	-4.24	0.00054	0.05
CP2 ($\mu = 80.81$), CP3 ($\mu = 83.20$)	-6.59	0.0000045	0.05
CP3 ($\mu = 83.20$), CP4 ($\mu = 83.48$)	-3.57	0.0023	0.05

Table 6.5: paired-sample ttests for stop criteria in improved method

For our proposed method without using auxiliary domain dropping, the test for CP3 and CP4 failed in rejecting null hypothesis, so there is no significant difference in their resultant accuracy, but for CP3 were needed 34.38 iterations on average, while 62.83 iterations were needed on average for the proposed method to finish. For the proposed method using auxiliary domain dropping every test rejected the null hypothesis.

6.5 Mind the gap between domains

We tried to measure the difficulty of the domain adaptation problem on each case, by determining how much the auxiliary domain is related to the

target domain. Text domains may hold a relation by sharing words in their vocabularies or by having similar distributions of their data points, in this case the documents.

Tables 5.4 and 5.11 in chapter 5 show the size of the vocabularies for each of the problems in the 20Newsgroups sets and the Amazon sets, respectively. The 20Newsgroups problems, comp_vs_rec, comp_vs_sci, and comp_vs_talk have the largest set of attributes if we take the two domains together, but they have smaller percentages of shared attributes than the other three problems, from 17.97% to 16.71%. Meanwhile rec_vs_sci, rec_vs_talk, and sci_vs_talk have fewer attributes, but they share a greater percentage. It is reasonable to think that with a bigger intersection between the attribute spaces, the auxiliary domain is closer to the target domain, and the accuracy of the baseline would be high. On the contrary by correlating the shared vocabulary with the baselines by using the pearson-moment correlation coefficient (eq. 6.1), results in $r = -0.4177$, $p = 0.4092$, the negative sign suggests a decreasing linear relationship. When the intersection of attribute sets is large, the performance of the classifier trained with the auxiliary domain on the target domain tends to be inferior. This seems to contradict the statement previously proposed, according to [15] the strength of the correlation is moderate, however by taking the thumb rule for a significant correlation $|r| > 2/\sqrt{n}$, the required r for 6 values is 0.8164, also the p value is far bigger than 0.05, so the correlation is not significant. The same test on the multi-sentiment data resulted on a strong positive correlation of $r = 0.7162$, $p = 0.0088$, which is significant.

The vocabularies were calculated after the stop-words removal and the correlation coefficient was defined as the following equation.

$$correlation = \frac{cov(x, y)}{\sigma_x \sigma_y} \tag{6.1}$$

The difference between two distributions can be measured by using a supervised classifier. We consider each domain as a class, then we label every instance as the class of the domain it belongs. If the classifier is able to separate the two distributions, then its performance is an indication on the distance between the two distributions [9] , 10-fold cross-validation was done.

The table 6.7 presents the closeness between domains. This value is the error of separating auxiliary and target distributions, the higher this value, the closer the domains.

#	Problem	Closeness between domains
1	comp_vs_rec	8.34
2	comp_vs_sci	10.89
3	comp_vs_talk	8.86
4	rec_vs_sci	4.73
5	rec_vs_talk	3.01
6	sci_vs_talk	4.98
7	electronics_vs_dvd	2.92
8	kitchen_vs_dvd	2.1
9	books_vs_electronics	1.77
10	dvd_vs_books	4.52
11	kitchen_vs_electronics	9.65
12	electronics_vs_books	1.72
13	books_vs_kitchen	2.1
14	kitchen_vs_books	2.15
15	dvd_vs_kitchen	2.12
16	dvd_vs_electronics	3.02
17	electronics_vs_kitchen	9.47
18	books_vs_dvd	4.45

Table 6.6: Closeness between target and auxiliary domain distributions

By correlating the closeness with the achieved accuracies with the proposed method, there might be a positive linear correlation $r = 0.5523$. The accuracies obtained with the proposed method with auxiliary domain dropping also seems positively correlated to the closeness of the domains $r = 0.5737$.

6.6 Experimental Observations

After obtaining all the experimental results, we calculated the pearson-moment correlation coefficient (eq. 6.1) over several of the variables in the problem; in order to look for possible correlations between them. The following subsections present those observations.

6.6.1 Other Observations in the Cross-domain thematic Experiments

There was a negative linear correlation between the accuracy of the baselines and the improvement in accuracy obtained by using the proposed method $r = -0.98$. So those cases with a low accuracy in their Baseline, had the highest improvement with the proposed method. We also correlated the percentage of shared vocabulary with the improvement in the accuracy. We observed that those cases with highest percentage of shared vocabulary between domains, obtained a higher improvement in their accuracy with the proposed method $r = 0.69$. Those cases with the lowest Baseline accuracy tended to reach the CP4 after a greater number of iterations $r = -0.85$.

6.6.2 Other Observations in the Cross-domain Sentiment Analysis Experiments

Consistently with the thematic case there was a negative linear correlation between the accuracy of the baselines and the improvement in accuracy obtained by using the proposed method $r = -0.79$. On the other hand, contrarily to cross-domain thematic experiments, cases with lowest percentage of shared vocabulary between domains, obtained a higher improvement in their accuracy with the proposed method $r = -0.78$. By correlating the in-domain accuracy, calculated as an upper-bound, with the final accuracy reached with the use of the proposed method $r = 0.84$. So the cases with a better in-domain classification obtained the best final results in accuracy.

Chapter 7

Conclusions and future work

7.1 Conclusions

The domain adaptation problem on the field of text classification deals with many special situations. First the potential feature space on text classification for a single domain might be huge; nevertheless the shared elements between domains might be very little and the useful knowledge for training an effective model for the new domain is limited. We observed that the approach of this thesis of gradually adding instances from the new distribution helps to improve gradually the accuracy of the classification. This happens because in this way the shared space among the domains becomes wider and then when it is reduced still is more descriptive of the target domain.

This work contributed with a domain adaptation method for text classification, useful in situations where labeled data are not available from the target domain, instead we have a set of labeled data from a related domain. The proposed method proved to perform well on the testing scenarios on this work. Compared with regular supervised learning, the proposed method considerably reduces the error rate on text classification. The proposed method competed fairly with other state of the art methods that address this problem. As the proposed method is self-adjusted to the changing distribution of the data during the process, it does not need the setting of magic parameters by the user, a frequent issue on the current domain adaptation method for text classification. The proposed method is conceptually simple and easy to implement making it a good option to treat real life problems effectively.

A new strategy to chose the instances of the test set in a self-training process for text classification problems with a good confidence is proposed. This work also presents an strategy for integrating the instances to the training set with a dynamic strict threshold.

We addressed the problem of a changing the feature space on a self training scenario, where in every iteration new documents are injected to the training set hence new vocabulary is injected. The feature space needs to be recalculated at each iteration. In this way the new knowledge added by this new vocabulary may be exploited.

In this work we tried to identify a measure to evaluate the feasibility for applying our method according to the gap between domains, but we could not obtain strong results on this objective.

The case of comp_vs_sci problem in the 20Newsgroups dataset is a very interesting case to analyze. comp_vs_sci auxiliary domain was the closest to its target domain according to the measure of closeness presented on section 6.5 among the 20Newsgroups problems. According to this we would think that it was an easier domain adaptation problem, and the results of supervised learning will be better than others, but it was actually the worse. In contrast this problem achieved the best error reduction.

Domain adaptation is possible to a certain extent. If the auxiliary domain is too different from the target domain the results might not improve much or they could go even worse.

7.2 Future work

The training set expansion with instances of target domain do not takes into consideration the balance of classes in the problem. The selection method just considers the fulfillment of the similarity above a threshold. The instance selection made might not represent all the classes, or it might favor only one class, as the iterative process continues the training set will become more unbalanced and the model trained with that training set would be more biased on each iteration.

The reliability of the training set expansion depends on the prototypes and their ability to represent dense population areas. In problems with a highly sparse data points, or the case when a lot of the instances are close to the margins of the separation hyperplane, there should be other ways to compute the prototypes which are more flexible and able to adapt to the distribution of the training set. In the near future we plan to test some other ways to compute the prototypes.

In order to adapt the training set to the target domain, not only adding target domain instances to the training set is feasible. We believe that removing the auxiliary domain instances from the training set during the iterative process, might also improve the ability of the model to deal with the target domain. A weighting scheme in the prototypes calculation might also be an option, in the near future we would like to try a weighting scheme to adjust the importance of instances dynamically every iteration.

The current approach intends to pick the instances far from the boundary defined by the separating hyperplane, by doing this, it is very unlikely for the linear model produced by the SVM classifier to change much, because the new instances in the training set might not define new support vectors.

Domain adaptation in text classification problems is highly related with the attribute spac, the shared features that are equally or similarly distributed in both domains. A more effective representation of those attributes in both domains might be tried.

Bibliography

[1] Amittai Axelrod, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 355–362, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[3] John Blitzer and Mark Dredze. Multi-domain sentiment dataset (version 2.0). `http://www.cs.jhu.edu/~mdredze/datasets/sentiment/`. Accessed: 2012-11-23, Posted: 2009-23-03.

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

[5] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 120–128, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[6] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, COLT' 98, pages 92–100, New York, NY, USA, 1998. ACM.

[7] Ondřej Bojar and Aleš Tamchyna. Improving translation model by monolingual data. In *Proceedings of the Sixth Workshop on Statisti-*

cal Machine Translation, WMT '11, pages 330–336, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[8] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[9] Laurent Candillier and Vincent Lemaire. Design and analysis of the nomao challenge - active learning in the real-world. In *Proceedings of the ALRA : Active Learning in Real-world Applications, Workshop ECML-PKDD 2012, Friday, September 28, 2012, Bristol, UK*, page to appear, 2012.

[10] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[11] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Transferring naive bayes classifiers for text classification. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 1*, AAAI'07, pages 540–545. AAAI Press, 2007.

[12] Hal Daumé, III and Daniel Marcu. Domain adaptation for statistical classifiers. *J. Artif. Int. Res.*, 26(1):101–126, May 2006.

[13] Hal Daume III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[14] Lixin Duan, Ivor W. Tsang, Dong Xu, and Tat-Seng Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 289–296, New York, NY, USA, 2009. ACM.

[15] J.D. Evans. *Straightforward Statistics for the Behavioral Science*. Psychology Series. Brooks/Cole, 1995.

[16] Arnulf B. A. Graf, Olivier Bousquet, Gunnar Rätsch, and Bernhard Schölkopf. Prototype classification: Insights from machine learning. *Neural Comput.*, 21(1):272–300, January 2009.

[17] Jeremy Greenfield. Ebooks account for 23 of publisher revenue in 2012, even as growth levels. *Digital Books World*, 2013.

[18] Hu Guan, Jingyu Zhou, and Minyi Guo. A class-feature-centroid classifier for text categorization. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 201–210, New York, NY, USA, 2009. ACM.

[19] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

[20] UCI Knowledge Discovery in Databases Archive. Uci knowledge discovery in databases archive: 20 newsgroups. `http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.data.html`. Accessed: 2012-11-23, Posted: 1999-09-09.

[21] Yang-Sheng Ji, Jia-Jun Chen, Gang Niu, Lin Shang, and Xin-Yu Dai. Transfer learning via multi-view principal component analysis. *Journal of Computer Science and Technology*, 26(1):81–98, 2011.

[22] Antonio Jimeno-Yepes and Alan R. Aronson. Self-training and co-training in biomedical word sense disambiguation. In *Proceedings of BioNLP 2011 Workshop*, BioNLP '11, pages 182–183, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[23] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 143–151, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[24] Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK, 1998. Springer-Verlag.

[25] Cristina Kadar and José Iria. Domain adaptation for text categorization by feature labeling. In *Proceedings of the 33rd European conference on Advances in information retrieval*, ECIR'11, pages 424–435, Berlin, Heidelberg, 2011. Springer-Verlag.

[26] Jyrki Kivinen and Manfred K. Warmuth. The perceptron algorithm vs. winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. In *Proceedings of the eighth annual conference on Computational learning theory*, COLT '95, pages 289–296, New York, NY, USA, 1995. ACM.

[27] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[28] PederOlesen Larsen and Markus Ins. The rate of growth in scientific publication and the decline in coverage provided by science citation index. *Scientometrics*, 84(3):575–603, 2010.

[29] James Lewis, Stephan Ossowski, Justin Hicks, Mounir Errami, and Harold R. Garner. Text similarity: an alternative way to search medline. *Bioinformatics*, 22(18):2298–2304, 2006.

[30] Qi Li. Literature survey: Domain adaptation algorithms for natural language processing. 2012.

[31] Yi Liu and Y.F. Zheng. One-against-all multi-class svm classification using reliability measures. In *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pages 849–854 vol. 2, 2005.

[32] Eddy Mayoraz and Ethem Alpaydn. Support vector machine for multiclass classification. In *IN PROC. OF INTERNATIONAL WORKSHOP ON ARTI NEURAL NETWORKS (IWANN'99*, 1999.

[33] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[34] Bernhard Pfahringer. A semi-supervised spam mail detector. In *In Proceedings of the ECMLPKDD Discovery Challenge Workshop*, page 48, 2006.

[35] Royal Pingdom. Internet 2012 in numbers. `http://royal.pingdom.com/2013/01/16/internet-2012-in-numbers/`. Accessed: 2013-04-29, Posted: 2013-01-16.

[36] Wen Pu, Ning Liu, Shuicheng Yan, Jun Yan, Kunqing Xie, and Zheng Chen. Local word bag model for text categorization. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 625–630, 2007.

[37] G. Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[38] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002.

[39] Internet Live Stats. Internet live stats. `http://www.internetlivestats.com/`. Accessed: 2013-11-23.

[40] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[41] Chang Wan, Rong Pan, and Jiefei Li. Bi-weighting domain adaptation for cross-language text classification. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1535–1540. AAAI Press, 2011.

[42] Ranks NL webmaster tools. Ranks nl default english stopword list. `http://www.ranks.nl/resources/stopwords.html`. Accessed: 2012-07-08.

[43] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, ACL '95, pages 189–196, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics.

[44] Di Zhang, Gui-Rong Xue, and Yong Yu. Iterative reinforcement cross-domain text classification. In *Proceedings of the 4th international conference on Advanced Data Mining and Applications*, ADMA '08, pages 282–293, Berlin, Heidelberg, 2008. Springer-Verlag.

[45] Yi Zhen and Chunping Li. Cross-domain knowledge transfer using semi-supervised classification. In *Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, AI '08, pages 362–371, Berlin, Heidelberg, 2008. Springer-Verlag.

[46] Yabin Zheng, Shaohua Teng, Zhiyuan Liu, and Maosong Sun. Text classification based on transfer learning and self-training. In *Natural Computation, 2008. ICNC '08. Fourth International Conference on*, volume 3, pages 363–367, 2008.

[47] Xiaojin Zhu and Andrew B. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan and Claypool, 2009.