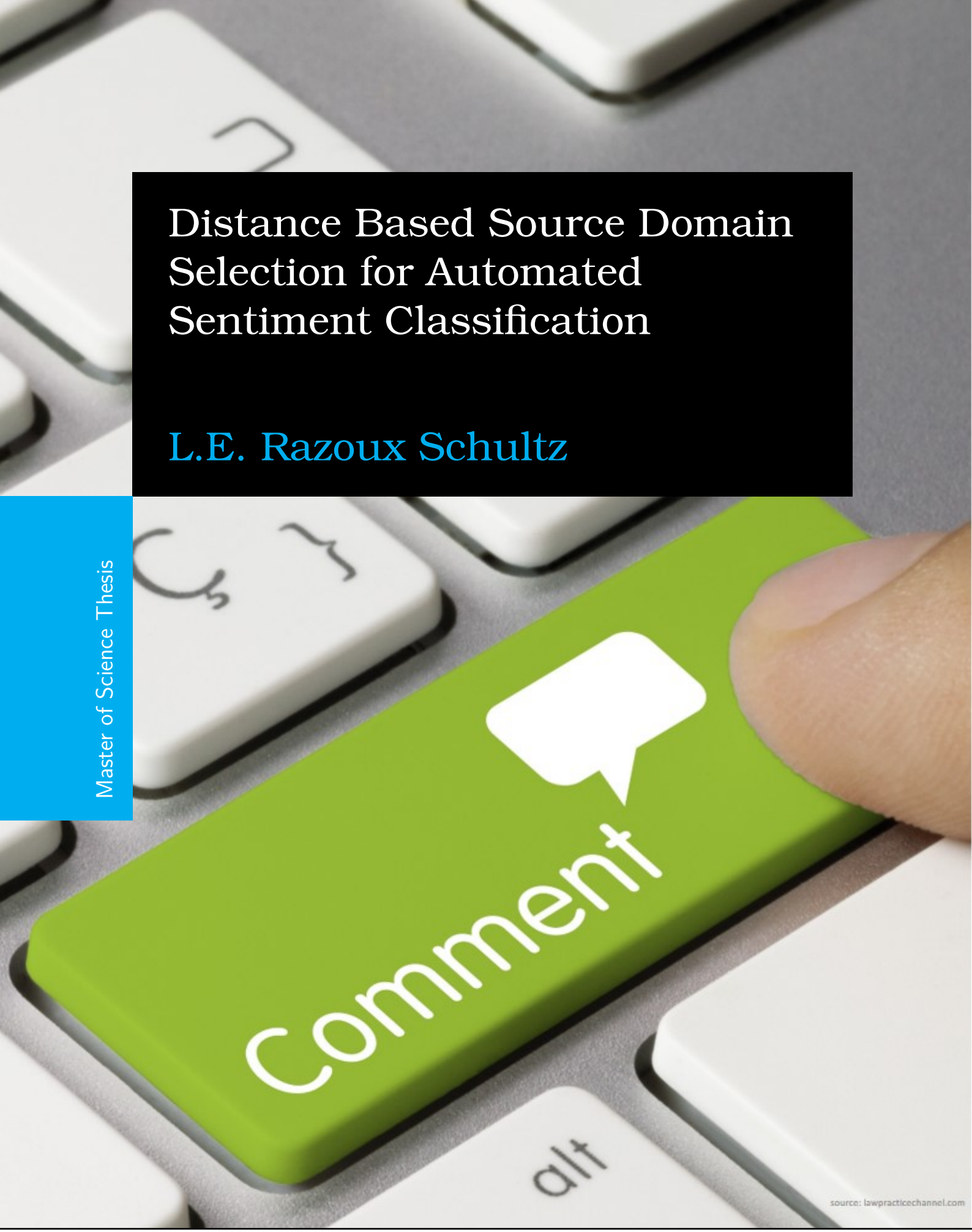
Distance Based Source Domain Selection for Automated Sentiment Classification

L.E. Razoux Schultz

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Distance Based Source Domain Selection for Automated Sentiment Classification

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

L.E. Razoux Schultz

March 20, 2018

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by Crunchr workforce reporting and people analytics software and the Pattern Recognition Laboratory Delft. Their cooperation is hereby gratefully acknowledged.

Abstract

Automated Sentiment Classification (SC) on short text fragments has been an upcoming field of research. Different machine learning techniques and word representation models have proven to be successful in classifying sentiment of opinion expressions in various domains, i.e. different topics or source media. However, when training on a source domain different from the target domain of interest, we encounter a large domain shift resulting in poor cross domain classification performance.

In this report, we first provide information on the key principles of SC, starting with the SC pipeline and the encountered domain shift. Then, we show a novel method of selecting a source domain by using four unsupervised distance measures: Chi squared (Chi2) distance, Maximum Mean Discrepancy (MMD), Earth Mover's Distance (EMD) and Kullback-Leibler Divergence (KLD). We evaluate the effectiveness of using these unsupervised measures individually, and in a linear combination, to identify one or more suitable source domains for an SC task for various target domains. This linear combination is proposed as the CMEK model, an acronym of the four measures it uses.

Results show that our proposed CMEK model for source domain selection results in a reduction of adaptation loss by 7 percent points compared to training on a randomly selected source domain. When selecting multiple domains, our proposed selection method is competitive with training on all data. In the light of general performance, we recommend the CMEK model for source domain selection for an SC task. The CMEK model shows significantly good performance and stable behavior in selecting multiple source domains and it has solid performance in selecting the single best domain.

Table of Contents

List of Figures

List of Tables

Preface

This document is the final deliverable of my Master of Science graduation thesis, summarizing the full graduation project study on source domain selection for automated sentiment classification. In first sight, the subject of automated sentiment classification does not seem naturally associated with the Master of Science program Systems and Control. However, the pipeline of performing sentiment analysis, including preprocessing, representation and classification together with identifying a suitable source domain, have certain parallels with system theory and system identification. In addition, future implementation of sentiment analysis will likely be in the field of robotics and man-machine systems.

The matter drew my attention and after a period of gaining more knowledge on machine learning techniques, I started researching the field of sentiment classification and wrote a literature survey on my findings. During this part of my research project, I came to find that there was a relatively unexposed problem. The effectiveness of sentiment classification was proven, but the effects on performance when training on the "wrong" data were not often described. This phenomenon known as domain shift seemed an interesting field of research with many opportunities to contribute and improve. In addition to methods such as transfer learning or domain adaption, I was intrigued by identifying a "good" data set for training a sentiment classification model. I was motivated to construct a model that could tell what would be the best domain to train on when we have data sets from different domains available. This motivation led to the development of a novel source domain selection model described in the end of this report and the resulting paper attached in Appendix A.

Acknowledgements

In the first place, I would like to thank my supervisor Peyman Mohajerin Esfahani for being a constructive discussion partner during my graduation project. In addition I would like to thank Marco Loog for his critical and expert view from a machine learning point of view.

At last I would like to thank Dirk Jonker and Rianne Kaptein from Crunchr for offering me an inspiring environment to work in and for offering the opportunity to develop useful knowledge by reflecting my findings on a "real world" problem.

Delft, University of Technology L.E. Razoux Schultz
March 20, 2018

Chapter 1

Introduction

This Master of Science thesis report serves as summary of the performed research in the light of the graduation project of the Master of Science Systems and Control from the Mechanical Engineering department at the Delft University of Technology. It serves as an autonomous deliverable including but not limited to findings from the previously performed literature survey and explanation of the developed model that is also described in the paper, see Appendix A. This study is supervised by the Delft Center for Systems and Control (DCSC) and the Pattern Recognition Laboratory (PRLab). The DCSC provides education and performs research in systems and control, where it contributes to fundamental aspects of dynamical systems and control as well as advancing innovative and high-tech applications [1]. Research topics include Robotics and mechatronics, Identification and estimation, Intelligent Control and Model-based control, among others [2]. The PRLab is concerned with the classical trinity of representation, generalization, and evaluation, being the core elements of every pattern recognition system. Examples of prominent research areas covered are dissimilarity-based pattern recognition, multiple classifier systems, and fields like semi-supervised and active learning [3].

The research topic of this study is improving automated sentiment classification (SC) of textual opinion expressions by developing a model for source domain selection. The task of SC in several distinguished domains is one that encounters aspects of identifying the domain of application and adapting accordingly as well as (semi-)supervised learning. Also, SC plays an important role in developing human-machine systems, since human sentiment has to be interpreted properly to construct a well performing system. Think of voice commands for example. The domain identification, machine learning approaches, and implementations in artificial intelligence and human-machine systems concerned with SC make this field of research interesting for both the DCSC and the PRlab.

The aim of this thesis report is to provide a concise description of the MSc. graduation project and the achieved results [4]. This will include presenting the proposed model as final contribution to the scientific community but also providing context and domain specific background knowledge. In addition, noteworthy findings and conclusions not related to source domain selection will be reported.

1-1 Report outline

Before reading, please be informed that this report can be dived into two parts. The first parts describes background research and findings in the general field of SC. This has been a major part in this master thesis project and is partly based on the preliminary literature study. This first part consists of chapter 2 and chapter 3. Then, we have a small bridge, chapter 4, to the second part describing the proposed method of selecting a source domain. This part includes chapter 5, chapter 6, the results, conclusion and discussion section. Based on the research described in this second part, a paper is written, see Appendix A. Both parts can be read separately. The second part mainly covers the content of the paper.

This report will start with providing background knowledge on the topic of SC. We point out its upcoming popularity and describe what Sentiment Analysis (SA) and, more specific, SC are. Next, we will give illustrative examples of current use of automated SC and sketch an idea of possible future applications that SC offers.

In chapter 3 we start by presenting a simple and clarifying example of how SC is performed. The pipeline for SC consists of three parts: preprocessing, representation and classification. Each part will be then looked at individually. For each part, we will present findings in literature and show which design choices can be made in making an SC model. To support conclusions, we constructed a model and tested different design choices on a data set containing employee survey reviews. Results will be reported.

Next, in chapter 4, we elaborate on the phenomenon domain shift that is deeply rooted in the field of SC. We report on the consequences of domain shift on the performance of classification and briefly elaborate on techniques that attempt to reduce the negative impact of domain shift on performance.

Then, we arrive at problem definition: how do we select the best source domain to train our SC model on for a given target domain? In chapter 5 we give a mathematical description of the problem and propose a source domain selection model that is able to select a suitable source domain from a set of candidates. The conceptual working of the model is supported on a couple of hypotheses that will be explained. The distance measures used in the model will be elaborated on.

To test the proposed model, we construct an experimental setup that will be explained in chapter 6. We elaborate on the used data sets, SC model design choices and statistical distance measures we use for our selection model.

We present the performance of the proposed source domain selection model, the CMEK model, and compare this with randomly selecting a source domain or training on all candidate source domains. We also look at the performance of the CMEK model when we let it select multiple domains to train on. In order to draw conclusions, significance of the results is tested and reported.

We conclude with interpreting the results of our experiments and generalizing this to a recommendation to use the CMEK model for source domain selection. We give a set of side marks and nuances with this recommendation, point out weaknesses of our model and offer suggestions for future work to improve the proposed CMEK model.

Chapter 2

Background

With the rapidly growing number of online reviews, comments and posts on products and services, the need for automated SC of textual opinion expressions arises [5]. Possible applications of automated SA lay in the field of recommendation systems [6], business intelligence, predictive models and survey response analysis. Using SA, companies are able to gain insights on big data opinion expression giving them information on topics such as customer satisfaction, investment opportunities and other business influential factors.

There are examples of companies providing services on big data SA, covering meta data on over 300 million public news websites, blogs, social media and public financial documents [7]. One can think of using instant SA of millions of sources in combination with topic detection as a useful tool for stock trading. Furthermore, recent studies claim that 90% of the people that read online reviews of a product were influenced by the content of the review in their purchase decision [8]. With over $300 billion of online purchases in the United States alone and almost 70% of the US population online shipping at least monthly [9], automated review SA qualifies as a potential goldmine. Also, claims have been made that SA on Twitter messages is a useful predictor for political matters such as the 2016 US presidential elections [10].

But, there are more future applications of SA that are some what further away but not less interesting. Think about the development of care robots that interact with elderly people to provide certain aspects of social needs. For the robot to interact properly, detecting the sentiment of expression of its human counter part is essential for serving its purpose. Noticing that a human is sad gives the robot a different angle of approach than when the human is cheerful. Thorough analysis of the sentiment of a human subject will have massive impact on man-machine communications. Other applications in this area are advanced voice command systems that induce more complex conclusions from speech than literally interpreting or executing voice commands. Think about how the reaction of chat robots for customer satisfaction departments can be improved by knowing what the sentiment of the respective customer is. And in other technological fields, is it possible for music streaming services to adapt the music you listen to your sentiment by analyzing what you type on your computer or smart phone?

Performing accurate SA is, however, subjective to certain environmental factors in the domain of implementation as will be discussed further in this report. Better and more accurate sentiment prediction will increase the value of the analysis, making it useful to research what factors influence the performance of classification and how to improve this performance.

2-1 SC as research field

SA can be seen as measuring the degree of presence of a certain evaluative subjectivity. Analysis can be performed on several aspects of subjectivity. It can measure presence of subjectivity of any kind but also more specific to emotions or evaluative topics such as angry, joyful, satisfied, positive or negative. The term 'sentiment analysis' is strongly related to umbrella terms such as semantic orientation or polarity, defined as an indication of the direction a word deviates from the norm for its semantic group or lexical field [11]. Also the terms subjectivity analysis or opinion mining are used, but a uniform terminology is not yet established [5].

In this report, we look at a more specific type of sentiment analysis, namely binary sentiment classification (SC). Binary SC refers to the specific binary determination of a textual document being of positive or negative sentiment; the task is to classify an expression in a positive or negative class.

Extensive research on machine learning techniques for the purpose of binary SC started in the early 2000's with papers from Pang et al. in 2002 [12] and Turney in 2002 [13], indicating this line of research is relatively new. In the following years, many papers have been published proposing techniques to improve performance of the classifiers proposed by Pang et al. and Turney, and applying them on new types of text messages such as Twitter messages. Different kind of preprocessing techniques have been evaluated and weighting schemes were introduced to give certain words more importance than others.

A new angle within SC was introduced by Google researchers Mikolov et al. in 2013 [14]. Their groundbreaking paper proposed an unsupervised algorithm building a lower dimensional feature space where distance between two points in this space is a measure of semantic equivalence. This research claims to observe large improvements in performance at much lower computational cost. However, to construct the embedding, it requires large amounts of data.

The data sets that are widely used mostly origin from move reviews data basis or product reviews. These sources are very convenient to use as they posses already some kind of information on the sentiment; reviewers give a rating to the movie or product. These ratings can be translated to positive and negative labels to obtain labeled data needed to train on. Other used sources are Twitter messages. This source is interesting as it contains sentiment information on a broad set of topics. It is often hand labeled or labeled based on emoticons the expressions contain.

Chapter 3

SC pipeline

The pipeline of SC consists of three parts: preprocessing, representation and classification. In this chapter, we first give a general outline of the process accompanied with an example and then dive into each of the three parts individually. In this deep dive, we will discuss model design parameters, findings in literature and results of own experiments.

3-1 General process of performing SC

Performing automated SC with supervised machine learning starts with acquiring labeled data points of the evaluated domain to construct a training set. These data points consists of a document d_i containing an opinion expression and a label. In case of binary SC, the labels either take the value positive or negative. The total set of documents, is called a corpus. A classifier is trained on this corpus and its labels and oughts to predict unlabeled data points, e.g. opinion expressions. Preferably, the training data is originated from the same domain as the unlabeled documents of which the sentiment has to be predicted.

From the acquired data, sentiment informative features have to be extracted. Most intuitive, is to choose words as features since we have textual data. But also combinations of words, part of speech, punctuation or even lengths or locations of words can be used as features. Recent research shows that also semantic properties of a word can be successfully used as features [14]. Choosing a set of n features, we build an n-dimensional space S in which each document d_i is a point which location is determined by the count of the chosen features present in the document.

Subsequently, a classifier is trained on these data points such that it constructs a hyperplane that optimally separates the negative labeled data points from the positive labeled ones by minimizing a cost function that penalizes misclassification. The hyperplane divides S in a positive class subspace S_+ and negative class subspace S_-, imposing $S_+^c = S_-$ and $S = S_+ \cup S_-$.

When sentiment of a new document has to be predicted, the document is positioned in S according to its feature values. If this location is in S_+, the predicted corresponding label is positive, if in S_-, it is predicted negative.

Let us illustrate these three steps with an example. The most basic, but still competitive, approaches in performing SC on text, are 'bag-of-word' approaches with a feature space consisting of (weighted) word counts as dimensions. Each text fragment is a point in this space; A document can be represented by a bag containing all the words of the document, disregarding grammar and word order but keeping multiplicity [15] and can be plotted as a point in the feature space. Our example follows the bag-of-words approach for binary SC. Assume we have the two documents containing both only one sentence

$$\begin{aligned}(d_1, POSITIVE): &\quad \text{“I like, really like my job”},\\ (d_2, NEGATIVE): &\quad \text{“I hate my job”}.\end{aligned}$$

We now choose to preprocess the data in a very minimal way. We only lower case all words and remove punctuation,

$$\begin{aligned}(d_1, POSITIVE): &\quad \text{“i like really like my job”},\\ (d_2, NEGATIVE): &\quad \text{“i hate my job”}.\end{aligned} \tag{3-1}$$

Next we choose a document representation model. Let us take all the verbs as features, resulting in a two dimensional feature space $S \in \mathbb{N}^2$ with the word “hate” on the x-axis and the word “like” on the y-axis. The value of a document in a single dimension is the count of the word corresponding to this dimension. The two documents in Eqn. (3-1) can be represented by two points in a two dimensional space

$$\begin{aligned}(d_1, POSITIVE): &\quad (x_1, y_1) = (0, 2),\\ (d_2, NEGATIVE): &\quad (x_2, y_2) = (1, 0).\end{aligned}$$

Now, for the last part, the classification, a hyperplane can be constructed to separate the positive points from the negative ones. To seperate the positive point from the negative point, we define the hyperplane as

$$y = 2x$$

implying

$$S_+ = \{x, y | y \geq 2x\}, S_- = \{x, y | y < 2x\}.$$

Now we need to test our model. Assume we have a test set containing again two documents with both again only one sentence

$$\begin{aligned}(d_3, \hat{y}_3): &\quad \text{“I like this car”},\\ (d_4, \hat{y}_4): &\quad \text{“I like, hate this car”}.\end{aligned}$$

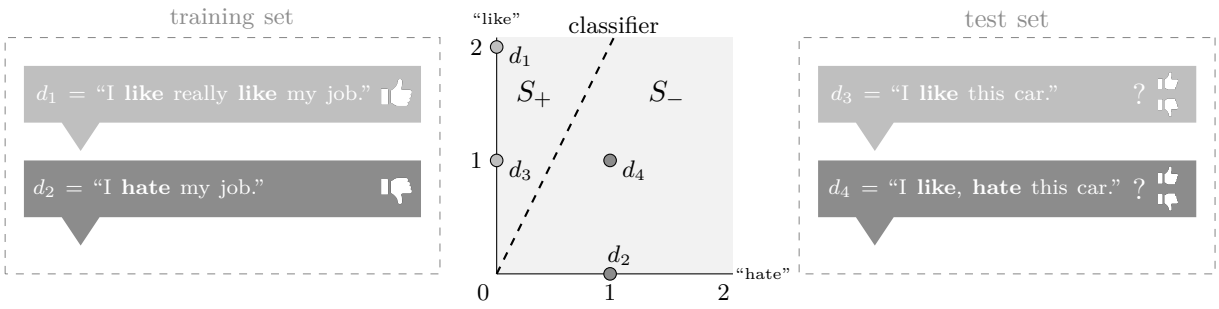


Figure 3-1: Example of binary two-dimensional linear classification with two single words as features.

Preprocessing these document and transforming them to points in S we have

$$\begin{aligned} d_3: &\quad (x_3, y_3) = (0, 1) \in S_+, \\ d_4: &\quad (x_4, y_4) = (1, 1) \in S_-. \end{aligned}$$

The classifier has classified d_3 to be of positive sentiment and d_4 to be of negative sentiment. The process described in this example is visualized in Figure 3-1.

For this simple example we see a classification error rate of 0 since we predicted the sentiment of all test documents correctly. We made some blunt choices on what preprocessing to do and how to represent the documents. Also, the training and test documents where very alike. In real world problems, the document are more diverse. In literature, we will typically see classification error rates between .1 and .3 when a classifier is trained on suitable data. To realize competing performance, we have to think about preprocessing, representation and classification more thoroughly than in our example.

3-2 Preprocessing

The input for the SC model, the training data, usually consists of documents containing raw text. If the training corpus consists of documents in multiple languages, it is preferable to filter documents on the language of interest first since different languages use different words to express sentiment. There are some simple preprocessing algorithms that can increase performance of classification by transforming the raw text of the training corpus. With this process we filter for certain type of words or create features other than the unique words.

3-2-1 Filtering

To reduce dimensionality, one can consider certain words as noise. These words do influence the training of the classifier but are assumed to hold no or only very little sentiment information. It can be considered to remove this noise by filtering some words in order to reduce dimensionality of the feature space. This improves computational efficiency and reduces overfitting. Stop words such as *the*, *a*, *of*, etc., intuitively have little to no sentiment information which raises the question if they have to be included in the features for SC. An

often used list of stopwords is the "Van stoplist" [16]. Note that negation words such as *no*, *never*, *nor*, etc., should not be removed as they hold much sentiment information, especially when used in combination with another word. Research suggests that the removal of stop words minimally affects the performance of classification [17]. Alternatively, when using an inverse frequency weighting of the features such as TF-IDF, the weight of stop words will automatically be lower due to the high frequency of stop words. Therefore, TF-IDF might be a good substitute for stop word removal. Furthermore, numerous researches use emoticons in documents to label sentiment of unlabeled documents [18] [19]. This suggests that these emoticons contain sentiment expression and can therefore be used as features for SA. Also, one can think of an exclamation mark in a sentence as a sign for higher probability of present sentiment in the sentence, for example.

3-2-2 N-grams

In some cases, sentiment is expressed by using several words. Frequently occurring positive class examples for product reviews are "I love", "love it" and "the best". For the negative class we see frequent occurrence of "returned it", "not worth" and "customer service" [20]. Notice that some of these expressions can be interpreted as different when looking at the individual words. Especially in case of negation words, it is important to look at the combination of words rather that analyzing each word separately. A set of N subsequent words is called an N-gram. The effect on performance of using N-grams as features with different values for N is not unambiguously described in literature. A survey on N-grams in SC states that using unigrams and bigrams is preferable over using N-grams with $N > 2$ [21]. But the conclusions of various papers differ, suggesting that the best choice of N is submissive to data characteristics. Own research on employee survey responses we acquired, shows that using $N = \{1, 2\}$ slightly beats using $N = 1$ or $N = \{1, 2, 3\}$, see Appendix B Figure B-1. A reason for improvement by using also bigrams as features could be the inclusion of some negations such as "not good" since using negation word combinations as features improves classification error [22].

3-2-3 Capturing semantic distinctions

Several studies elaborate on using a word plus its Part of Speech (POS) as feature and using the combination of the POS and the word as feature. Using POS improves the performance of SC when using unigrams and a bigrams of a word plus a part of speech [23]. Also, tagging each unigram feature with its POS, making distinction between different usage of same words, improves results [12]. For example, using POS tagging, there will be two features of the word "love" for documents "A love story." and "I love the movie." instead of one. POS tagging makes distinction between meaning of textual equivalent expressions.

As opposite of POS-tagging, we can treat different textual expressions as being of similar meaning. For some Natural Language Processing (NLP) implementations, different forms of a word may be treated the same to improve results. Two different techniques can be distinguished: stemming and lemmatization. Compared to lemmatization, stemming is a more crude and heuristic process of removing the derivational affixes of a word. Lemmatization is a process that usually uses a vocabulary and morphological analysis to transform a word

to it base form, called the lemma. Both techniques are used in NLP to write related forms of a word to their common base form [24]. In SC these techniques might be used to reduce dimensionality.

Applying stemming with the widely used Porter algorithm [25] will transform

"I am loving my job and colleagues, they make me happy!"

to

*"I am **love** my job and **colleagu**, they make me **happi!**".*

In information retrieval, using stemming techniques improves results of database search queries [26] [27]. There is, however, research suggesting that the effectiveness of stemming for English query systems is rather limited [28]. For SC, stemming and lemmatization seems to increase error [29]. Own experiments support the conclusion concerning lemmatization, see Appendix B Figure B-3.

There are various other preprocessing techniques. One, almost always used unmentioned technique, is writing all words in lower case so that a certain word at the beginning of a sentence (with a capital) is treated the same as the lower case word. One could also consider dividing a compound such as *couldn't* into *could* and *not*. Sometimes multiple characters are incorrectly used which creates a non existing word such as *hellllooooooo* or misspelled words. There are simple algorithms similar to spell checkers that transform these words into their expected correct word. The implications of these techniques on the error of classification are hard to extract from literature. There are, however, studies that construct algorithms that normalize 'social media' English which is proven to, at least, improve the correct spelling of words [30].

3-3 Representation

Most classifiers need numerical values as input in order to be trained and in order to predict labels. When performing SC, we initially have sequences of words as data and thus there is a need to transform these words into numerical values. We seek a way of transforming each word into a set of numerical values. This set of values can be represented as a vector or point in a feature space. In this section, we start by explaining the feature space and define it mathematically. To reduce dimensions or filter noise, we often use feature selection. We quickly discuss in what ways one can perform feature selection and introduce the concept of feature weighing. A number of different weighing schemes will be explained. Then, we will elaborate briefly on two popular distinct types of feature spaces to represent words; A high dimensional space where all words and N-grams correspond to a single dimension, and a word embedding where this high dimensional space is transformed into a lower dimensional space. This transformed space has some very interesting properties for NLP.

3-3-1 Feature space

Let us start off with defining some of the concepts earlier mentioned, namely a feature, a feature space and a class. We begin with a simple approach in which we will only use

$$d_i \subset \wp(W) = \left\{ \begin{array}{c|c|c} & \overbrace{\quad\quad\quad\quad}^{\mathcal{F} \subset \wp(W)} & \\ D_{tr} & f_1 \quad f_2 \quad \cdots \quad f_{n_F} & Y_{tr} \\ \hline d_1 & 1 \quad 0 \quad \cdots \quad 0 & y_1 \\ d_2 & 0 \quad 0 \quad\quad\; 2 & y_2 \\ \vdots & \vdots \quad\quad \ddots \quad \vdots & \vdots \\ d_{n_{tr}} & \underbrace{0 \quad 3 \quad \cdots \quad 1}_{X \in \mathbb{N}^{n_{tr} \times n_F}} & y_{n_{tr}} \end{array} \right\} = y_i \in \{0, 1\}$$

Figure 3-2: Feature space representation of training data set.

single words to classify a document, we do not use N-grams, lemmatization etc. In this scenario, we start by defining a vocabulary W that is a finite collection of n unique words, $W = \{w_1, w_2, ..., w_n\}$ and $|W| = n$. For simplicity we can assume that the vocabulary exists of all unique words in the corpus. Next, from the vocabulary, we define a feature set $\mathcal{F}$ which is a subset of the power set of all words denoted as $\wp(W)$ which represents all possible combinations of words from W, $\mathcal{F} \subset \wp(W)$ and $|\mathcal{F}| = n_F$. The process of choosing $\mathcal{F}$ from W is referred to as feature selection.

To build and test a classifier, we have a training set and a test set. These sets consist of tuples each containing a document d_i and its corresponding label y_i, in our case we have the label values 0 and 1 corresponding to the, respectively, negative sentiment class and positive sentiment class. Note that each document is actually a set supported on the vocabulary as it is a combination of words of the vocabulary. The set of all possible documents is denoted as $\wp(W)$ and $d_i \in \wp(W)$. Each tuple (d_i, y_i) is referred to as a labeled object. The training set consists of n_{tr} labeled objects and the test set consists of n_{ts} objects. So, the training and test set can be represented by, respectively n_{tr} and n_{ts} objects (d_i, y_i) where $d_i \in \wp(W)$ and $y_i \in \{0, 1\}$.

Now, each document d_i is converted into an n_F-dimensional feature vector x_i by a count vectorizer Z. This count vectorizer Z simply counts the occurrences of each feature in the inputted document and outputs a vector of counts, $Z(d_i) = x_i \in \mathbb{N}^{n_F}$, where $\mathbb{N}^{n_F}$ is called the feature space. When the entire set of documents from the training set $D_{tr} = \{d_1, d_2, ..., d_{n_{tr}}\}$ is inputted in Z we get $Z(D_{tr}) = X_{tr} \in \mathbb{N}^{n_{tr} \times n_F}$, where X_{tr} is referred to as our feature values matrix of the training data. In our evaluated domains, documents typically have relatively few words, resulting in a highly sparse matrix X_{tr}. With the similar transformation Z for the test data set D_{ts} we get $Z(D_{ts}) = X_{ts} \in \mathbb{N}^{n_{ts} \times n_F}$. The set of labels of the training and test documents are denoted as Y_{tr} and Y_{ts} respectively.

In many cases, preprocessing is performed, resulting in a transformation of the feature set $\mathcal{F}$ by transforming words from the documents (lemmatization, stemming, spelling correction and lower casing), by removing words from the documents (stop word removal or punctuation removal) or by creating new words or entities (N-grams and POS-tagging). Often a transformation of the feature matrix X is performed to weight the features or to transform it into a lower dimensional continuous feature space, as is discussed later on in the setion.

3-3-2 Feature selection

In a traditional features space, all words and N-grams are included in the feature set $\mathcal{F}$, resulting in a very high dimensional space, typically $n_F >> 10^4$. However, a lot of these words and N-grams do not carry much sentiment or no sentiment at all. More precisely, it is not possible for the classifier to classify sentiment based on these features. Intuitively, words such as *the*, *of* or *by* do not tell much about the sentiment in the document. And what about the words *walking*, *tree* or *boss*? A lot of words can be seen as noise and can be filtered to reduce dimensionality. If we are interested in using only a subset of all words and N-grams in the corpus we perform feature selection to define a subset of the initial feature space $\mathcal{F}' \subseteq \mathcal{F}$. In this section we will look into the effects of performing feature selection on the classification error. We take a closer look at two popular feature selection methods: Mutual Information (MI) and χ^2-selection.

MI selection is a method that selects "the most informative" features by measuring the mutual dependence between a feature and a class. It measures the amount of information of a class given feature presence and vica-versa. The MI score can be calculated according to

$$MI(F;C) = \sum_{e_f \in \{0,1\}} \sum_{e_c \in \{0,1\}} P(F = e_f, C = e_c) log_2 \left(\frac{P(F = e_f, C = e_c)}{P(F = e_f)P(C = e_c)} \right)$$

where F is random variable that takes value e_f= 0 if the feature is not present and takes the value $e_f = 1$ if the feature is present, C is a random variable taking value e_c= 0 if the class is negative and $e_c = 1$ when the class is positive [24]. Note that when the feature F is not correlated with the class C, the log function becomes 0 and the MI-score will be 0, indicating that there is no mutual information of F and C. For an example, see Appendix C. The $n_{F'}$ features with the highest MI score will be selected. Research indicates that for opinion detection, merely if there is an opinion or not, the MI method does not significantly improves the results using a relatively small data set [31]. They note that this can be due to class imbalance. However, for a categorization task of small text fragments, it does improve the error of classification significantly. In an SC task of 2000 movie reviews, slight improvements are realized when using MI selection but no statistical significance is mentioned [32].

Another selection method is χ^2-selection. This method first calculates the expected number of objects with $(F = e_f, C = e_c)$ when drawing from a mixed set with both classes. These expected values are denoted by E_{e_f,e_c}. These values are compared with the actual number of drawings denoted by N_{e_f,e_c}. The χ^2-value is calculated [24] as

$$\chi^2(F,C) = \sum_{e_f \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{\left(N_{e_t e_c} - E_{e_t e_c}\right)^2}{E_{e_t e_c}}.$$

For an example, see Appendix C. Note that when there is no correlation between the presence of a feature F and a class C, the χ^2-value is zero. The feature with the $n_{F'}$ highest χ^2-values get selected to build a feature space. Research indicates that the χ^2 method for feature selection improves results for short text classification [31]. Other research that tries to improve SC in the movie review domain with again 2000 documents shows improvement in performance but statistical significance is not mentioned [32]. Own experiments suggest that

we can use a χ^2-selected subset of the vocabulary of unigrams and bigrams and get similar performance as using all unigrams and bigrams. However, χ^2-selection will not increase performance. This conclusion seems not subjected to the number of training samples. See Appendix B for support for this conclusion.

3-3-3 Feature transformation

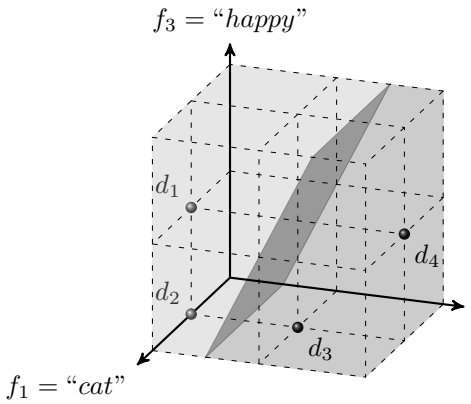
To summarize the process so far, we started with a set of n_{tr} documents d_i and their corresponding binary labels y_i. From the documents we extract all unique words and call this set our vocabulary W. By performing text preprocessing on all documents, the set of words in the vocabulary will change. Common preprocessing techniques are using N-grams, lemmatization, lower casing all words or performing spelling correction. The changed vocabulary we now call our initial feature set $\mathcal{F}$. After feature selection we remain with a subset of our initial feature space $\mathcal{F}' \subseteq \mathcal{F}$. A regular count vectorizer Z counts the occurrences of all features in a document and outputs a vector of counts. Performing this operation for all documents results in a high dimensional, sparse and discrete feature value matrix $X_{tr} \in \mathbb{N}^{n_{tr} \times n_{F'}}$ and $X_{ts} \in \mathbb{N}^{n_{ts} \times n_{F'}}$. Now, we have the possibility to transform these matrices by giving certain words more weight.

The elements of X_{tr} and X_{ts} are called the raw term frequencies and the term frequency of the feature f_j in document d_i is denoted as tf_{ij}. These elements are called raw term frequencies since they represent the actual count of features in a document. Instead of counting the number of occurrences of a feature in a document, one can also just look at the presence of a feature. If the feature f_j is present in the document d_i, the element of the feature value matrix x_{ij} gets a value of 1 and 0 if it is absent. See Figure 3-3 and 3-4 for an example of building a feature space and transforming it according the presence based approach. Research shows that the presence based approach is preferable over the raw term frequency for SC [12]. This is in contradiction to the task of topic classification [12] [33] [34]. For the movie reviews data set used by Pang et al. we find, using the raw term frequency, an classification error of 16.60% and for presence based (boolean or binary weighting) we have an error of 12.15% [33]. Most research seem not to use the presence based approach but to use feature weighing.

A widely used technique for weighting features is the Term Frequency-Inverse Document Frequency (TF-IDF) weighting formula, originated from information retrieval systems. The feature value for feature f_j of a document d_i is calculated as a function of the number of occurrences of f_j in d_i, called the term frequency tf_{ij}, the total number of documents $N = n_{tr} + n_{ts}$ and the number of documents in which f_j is present, n_{f_j},

$$x_{ij} = tf_{ij} \times log\left(\frac{N}{n_{f_j}}\right).$$

The base of the logarithmic function can be used as parameter. This weight function can be seen as base line weight function as it is used in 83% of all text-based recommender systems [35]. Note that if a term occurs in every document, the weighting function assumes this term is of no relevance for SC and therefore will relinquish the corresponding feature. Often, for a two class problem, we can integrate a class distinction using the formula



w_1 = beautiful
w_2 = cat
w_3 = happy
w_4 = I
w_5 = is
w_6 = made
w_7 = me
w_8 = sad
w_9 = the
w_{10} = was
$\mathcal{F} = \{w_1, w_2, w_8\}$

d_1 = "The **cat** makes me **happy**."	POS
d_2 = "The **cat** is beautiful."	POS
d_3 = "The **cat** makes me **sad**."	NEG
d_4 = "I was **happy**, the **sad cat** made me **sad**."	NEG

Figure 3-3: Vocabulary, feature set and feature space for example data set.

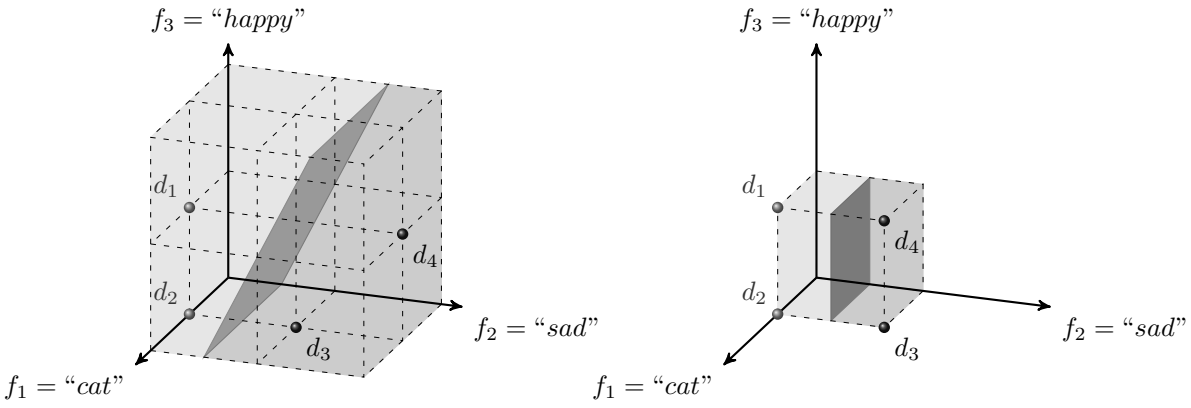
$$x_{ij} = tf_{ij} \times log\left(\frac{N_0 \times n_{f_j,1}}{N_1 \times n_{f_j,0}}\right)$$

where the subscripts 0 and 1 denote the class label. For example, N_0 is the number of documents with label 0. This type of Inverse Document Frequency (IDF), we call Delta IDF. On the same movie review data set as before (Pang et al.), we see a classification error of 8.40% for this TF-IDF weighting function with class distinction [33].

3-3-4 Embedded feature space

The high dimensional and sparse feature value matrix X_{tr} has some disadvantages due to its high dimensionality. In the early 2000's, approaches to transform the high dimensional and sparse matrix X_{tr} into a lower dimensional matrix where developed which opened a whole new world for NLP, including SC.

In 2003 Y. Bengio proposed a neural probabilistic language model to learn a joint probability function of sequences of words [36]. This model uses a feature transform of each word into a lower dimensional space as input for a neural network in order to predict the joint probability function of sequences of words. The mapping of a space where each dimension represents the count of a unique word into a lower dimensional continuous vector space, typically 50-1000 dimensions, is called a word embedding [37]. The proposed model therefore uses an embedded feature space to represent words. One of the interesting properties of this representation of words is, that similar words are expected to have a similar feature vector [36]. This property can be used to distinguish words or documents containing positive sentiment information from words or documents containing negative sentiment information [38]. Further improvements on Bengio's model have led to the popular word2vec algorithm [14] and doc2vec algorithm [39].



(a) Raw term frequency feature space.

(b) Presence of term feature space.

Figure 3-4: Feature space transformation from raw term frequency to term presence.

The construction of the embedding requires much training data. Since most of the acquired data sets in our experiments are not large enough to construct a proper embedding, we will not use an embedded space. Combining multiple data sets to build an embedding would compromises the purity of conclusions since results might be affected by the embedding that is constructed by data that is not related to the individual data set under evaluation. For further reading on the embedded feature space, see Appendix D.

3-4 Classification

With all n_{tr} labeled document in the training set defined in the selected feature space $\mathcal{F}$, we now try to find a relation between the location of each document d_i in $\mathcal{F}$, or the feature values of d_i, and its label y_i. This approach is called supervised learning due to the fact that it uses labeled documents for training instead of unlabeled ones, in which case it would be unsupervised learning. Now, we denote the discrete space S in which our feature values matrices exist as the discrete feature space spanned by $\mathcal{F}$, $\mathbb{X} \in \mathbb{R}^{n_F}$. The space in which the binary class labels exist is denoted as $\mathbb{Y} \in \{0, 1\}$. For this supervised learning task, we attempt to find the true hypothesis function $h^\star : \mathbb{X} \to \mathbb{Y}$ that maps the feature value vector $x_i \in \mathbb{X}$ to its corresponding label $y_i \in \mathbb{Y}$. Recall that in our binary sentiment classification task, $\mathbb{Y}$ merely consists of the set $\{0, 1\}$.

To restrict the search for the true mapping $h^\star$ from all, infinite many, possible mappings, we restrict the search to a limited family of candidate functions $\mathcal{H}$. We will further refer to $h \in \mathcal{H}$ as a candidate hypothesis function and $\mathcal{H}$ as the hypothesis space. In our classification task, we have to select a learning algorithm that constructs the optimal hypothesis function $\hat{h} \in \mathcal{H}$ that approximates $h^\star$. Often, in classification, a threshold is used to determine class

$$y_i = \begin{cases} 1 & \hat{h}(x_i) > 0 \\ 0 & \text{else.} \end{cases}$$

In order to be able to define what the optimal hypothesis is, a loss function ℓ is chosen which has to be minimized. The loss function typically reflects the mismatch between the predicted

output $\hat{y}_i$ by $\hat{h}(x_i)$ and the actual output y_i. The search for $\hat{h}$ is formalized by the following optimization program

$$\hat{h} := \arg\min_{h \in \mathcal{H}} \quad \sum_{i=1}^{n_{tr}} \ell(h(x_i), y_i)$$

where $x_1, x_2, \ldots, x_{n_{tr}}$ represent the feature values of the training objects and $y_1, y_2, \ldots, y_{n_{tr}}$ the corresponding labels. The optimal hypothesis function $\hat{h}$ is in machine learning referred to as trained classifier. To measure the performance of the trained classifier $\hat{h}$, we use the classification error defined as

$$\xi(\hat{h}, (X_{ts}, Y_{ts})) = \frac{1}{n_{ts}} \sum_{i}^{n_{ts}} \mathbb{1}_{\left\{\hat{h}(x_i) \neq y_i\right\}} \tag{3-2}$$

where X_{ts} are the feature values of the test set and Y_{ts} the corresponding labels. For proper performance assessment, no data from the test set is used for training.

In this chapter different $\mathcal{H}$ and ℓ will be explained and evaluated on performance for an SC task using a high dimensional feature space. Due to the high dimensionality, to prevent over fitting, only linear classifiers are considered. For orientation what would be suitable classifiers we did experiments. The best performing classifiers are Ridge Regression Classification, Logistic Regression Classification, Multinomial Naive Bayes and a linear Support Vector Machine, see Appendix B. These four classifiers are widely used and popular in performing SC. Therefore, these four classifiers will be explained in the upcoming part of this section.

3-4-1 Regression classifiers

Regression classifiers belong to the type of probabilistic classifier since they have the ability to predict the probability of an object belonging to a certain class rather than identifying only the most likely class it belongs to. To apply simple linear regression to classify points in the feature space, one assumes that the label is determined by a linear combination of its corresponding feature values with an additional error ϵ that is standard normal distributed, $\epsilon_i \sim \mathcal{N}(0, \sigma)$,

$$y_i = x_i \hat{\beta} + \epsilon_i, \quad i = 1, \ldots, n_{tr}$$

with $\hat{\beta}^T \in \mathbb{R}^{n_F}$, or in matrix form

$$y = X\hat{\beta} + \epsilon, \quad i = 1, \ldots, n_{tr}.$$

The model assumes linear independence of the objects. Now, the error ϵ represents the difference between the actual value of y and the prediction by the model $\hat{y}= X\beta$. The described model has some interesting properties. By maximizing the log-likelihood, the optimization problem boils down to the minimization of a least square loss function, which is convex and has a closed form solution. Therefore, optimization is generally fast. The minimization problem and its closed form solution are

$$
\begin{aligned}
y &\sim N(\beta x, \sigma^2) \\
\hat{\beta} &= \arg\max_{\beta} \mathcal{L}(x|y) \\
&= (X^T X)^{-1} X^T y
\end{aligned}
$$

inferring the hypothesis function

$$
\hat{h} : y_i = \begin{cases} 1 & x_i \hat{\beta} > 0 \\ 0 & \text{else.} \end{cases}.
$$

See Appendix C for the derivation.

Since our feature value matrix is usually rather sparse, we have a so called *ill posed* problem. For this type of problems, adding a regularization term to the loss function will improve classification performance. The regulatory term prevents the weights β to cluster onto one or a few features, it forces to spread the weight over the features. A proposed way to use this regularization, is to use the loss function

$$
\ell(y, X) = ||y - X\beta||^2 + ||\Gamma\beta||^2
$$

where Γ is often chosen as αI with I as the identity matrix and α a parameter to determine the intensity of regularization. A classifier with this loss function is referred to as Ridge Classifier. The analytic solution becomes

$$
\hat{\beta} = (X^T X + \Gamma^T \Gamma)^{-1} X^T y.
$$

A variation of simple linear regression classification that is commonly used, is Logistic Regression. This type of regression classification assumes Y not to be normally distributed around 0 but assumes Y to be distributed according to a logistic distribution centered around 0. The model assumes discrete output, in our case $Y \in \{0, 1\}$. This distribution is characterized by the function

$$
P(y = 1|x) = \frac{1}{1 + e^{-\beta x}}, \qquad P(y = 0|x) = \frac{1}{1 + e^{\beta x}}
$$

and the hypothesis

$$
\hat{h} : y_i = \begin{cases} 1 & P(y = 1|x) > 0.5 \text{ or } \hat{\beta} x > 0 \\ 0 & \text{else.} \end{cases}
$$

Now, maximizing the likelihood gives

$$
\hat{\beta} = \arg\max_{\beta}, \sum_{i=1}^{n_{tr}} y_i \beta x - \log\left(1 + e^{\beta x}\right).
$$

See Appendix C for the derivation. The solution has no closed form. As can be noticed, the logistic regression and normal regression classification have much in common. The differences are that logistic regression assumes y to be discretely distributed according to the logit function as cumulative distribution function. The normal linear regression model assumes y to be normally distributed. In contrary to normal regression classification, the logistic regression classification does not yield a closed form solution.

Pang et al. reviewed this classifier for a sentiment classification task on movie reviews and found that for most feature settings, the logistic regression classifier did a slightly inferior job compared to the Support Vector Machine (SVM) and Naive Bayes classifiers [12]. Own experiments show different results, see Appendix B.

3-4-2 Naive Bayes

The Naive Bayes (NB) classifier is a probabilistic classifier assuming independence of the features. In text classification, it is competitive to more advanced methods such as SVM's [40]. This classifier is one of the baseline methods for text categorization and SC and has been studied for decades. The benefits of this classifier are that it is computational fast as the number of parameters to be estimated scales linear with the number of variables. The NB classifier is based on the Bayes' theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

which can be translated to our specific classification task terminology

$$P(y_i|x_i) = \frac{P(x_i|y_i)P(y_i)}{P(x_i)} = \frac{\prod_{j=1}^{n_F} \left\{ P(x_{i,j} > 0|y_i)^{x_{i,j}} \right\} P(y_i)}{P(x_i)}.$$

The NB classifier does not have a loss function and the hypothesis $\hat{h}$ is constructed as

$$\hat{h} : y_i = \begin{cases} 1 & P(y_i = 1|x_i) > P(y_i = 0|x_i) \\ 0 & \text{else.} \end{cases}.$$

Research shows that the NB approach is inferior to the SVM [20] [41]. This conclusion is supported by own experiments, see Appendix B. Note worthy is that NB is computational cheap which can be of relevance for large SC tasks.

3-4-3 Support Vector Machines

Support Vector Machines (SVM's) are distance based classifiers since they do not predict a probability of an object belonging to a class. As measure for likeliness of two objects belonging to the same class, a distance metric is used, often the Euclidean distance. Other examples of distance based classifiers are k-Nearest Neighbour and Nearest Mean classifiers. An SVM classifier defines a hyperplane in the feature space separating positive class points

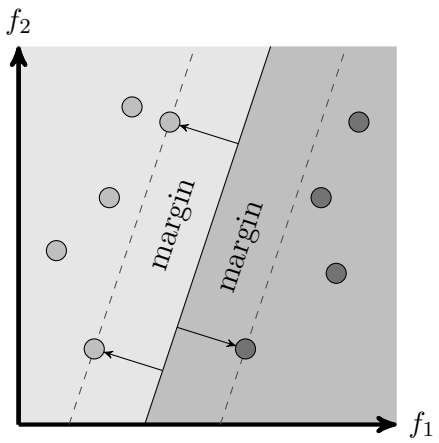


Figure 3-5: Two-dimensional SVM with corresponding margins.

from negative class points. The margin between the closest (Euclidean distance) points of both classes to the hyperplane is maximized. Development of SVM's started in the 60's and it is said that the first SVM techniques were recognized to be published in 1979 by Vladimir N. Vapnik [42] [43]. SVM's can be categorized into two approaches depending on linear separability of the object representations X. When X is linearly separable, a hard margin is used. Otherwise we use a soft-margin.

In many cases, especially for small data sets, the number of features is higher than the number of training object. In this case, it is likely that X is linearly separable, making it possible to construct a hyperplane that separates the objects with no error. In this case we do not have to define a loss function to penalize incorrect classification within the training set. Let us define two hyperplanes describing the margins, see also Figure 3-5,

$$wx - b = -1 \quad \text{and} \quad wx - b = 1.$$

Now, the distance between these margins has to be maximized. We can describe the optimization problem in constructing an SVM as

$$\begin{aligned} \hat{w}, \hat{b} = \underset{w,b}{\operatorname{argmin}} \quad & ||w|| \\ \text{subject to} \quad & wx_i - b \geq 1 \quad \text{if} \quad y_i = 1 \quad \text{and} \\ & wx_i - b \leq -1 \quad \text{if} \quad y_i = 0 \quad \text{for} \quad i = 1, \ldots, n_{tr}. \end{aligned}$$

And the hypothesis becomes

$$\hat{h} : y_i = \begin{cases} 1 & \hat{w}x_i - \hat{b} > 0 \\ 0 & \text{else.} \end{cases}$$

In case the data is not linearly separable, we have to define a loss function to penalize incorrect classified points. We then have a soft-margin. Often the hinge loss is used, giving the following loss function ℓ

$$\ell(x_i) = \begin{cases} \max(0, 1 - wx_i + b) & y_i = 1 \\ \max(0, 1 + wx_i - b) & y_i = -1. \end{cases}$$

The optimal parameter values for the hypothesis function are now defined by the optimization problem

$$\hat{w}, \hat{b} = \underset{w,b}{\operatorname{argmin}} \quad \left[\frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \ell(x_i) \right] + \lambda ||w||^2.$$

SVM's are widely used in SC. If we compare the results of Pang et al. for the NB and SVM classifiers, we see that for most feature sets, the SVM performs better [12]. For the best feature set, presence based unigrams and bigrams, the SVM also gives the lowest classifcation error. Same holds for results of Dave et al. [20]. The error on the data set of Kalaivani et al. are over the set $n_{tr} = \{100, 200, 300, 400, 800, 1600, 2000\}$ always in favour of the SVM classifier [41]. On average the SVM approaches outperforms the NB approaches. However, in most cases only slightly better than the best performing NB classifier.

Chapter 4

Domain shift

In supervised machine learning problems, we are interested in classifying certain objects of interest. In order to do so, we use labeled objects to train on. The data we train on originates from the source domain and the data we ought to classify originates from the target domain. Often, we assume the source and target domain to be of such similar nature that the characteristics of the samples from the two domains are the same. Mathematically speaking, the underlying distributions of the source sample and target sample are similar and therefore the expected samples are similar. In many cases, this similarity assumption is valid, but in other cases, the similarity assumption doesn't hold and classification performance is rather bad.

For example, a classifier is trained on Dutch citizens to diagnose diabetes based on Hemoglobin A1c level. The classifier proves to be successful. If we would use the same classifier to diagnose diabetes for Belgian citizens, we seem right to assume that the classifier will still perform well since Dutch and Belgian citizens live under similar circumstances. Their Hemoglobin A1c levels are assumed to be distributed similar and the similarity assumption holds. Now imagine that a US hospital asks to deploy the same constructed classifier in order to diagnose their American patients on diabetes. The classifier is used and we see a performance drop. It appears that the normal versus abnormal levels differ severely between Dutch and US citizens. In this case the underlying distributions are not equal and the similarity assumption does not hold, we have encountered a domain shift [44]. In other words, for optimal performance, the source domain and target domain have to be the same [45].

In this chapter, we will first state what we define as a unique domain. We will introduce the concept of adaptation loss to measure the extent of domain shift. Then, we will give proof of domain shift affecting the classification performance drastically in the case study of SC. We will briefly discuss techniques in literature that attempt to overcome the domain shift. We conclude this chapter by stating the relevance of proper source domain selection.

4-1 Definitions

We define the term 'domain' as a class of documents that follow same rules of sentiment expression. In the light of SC, a general domain view distinguishes domains such as product reviews, (professional) movie reviews and common textual communication such as email. A more detailed view will distinguish each unique person in each unique situation as a separate domain since opinion expression arose under different circumstances and is therefore assumed to be different in its nature, i.e. different in its distribution. The most strict definition of domain will state that documents share domain if and only if they are samples from the same underlying distribution. However, in practice we do not know this distribution.

We have to give a subjective definition of a domain to be able to evaluate them, we have to assume what kind of data has the same or somehow similar distribution. For generalizability, we give the following definition to a domain: Two documents share domain if they are retrieved from the same medium and targeting the same class of topics. Therefore, two movie reviews retrieved from the same medium but written by two different persons on two different movies are from the same domain. Documents retrieved from two different sources (websites) are not in the same domain. Tweets about politics and tweets about a sports game are not from the same domain, but tweets about mobile phone A and mobile phone B are within the same domain.

Strictly speaking, we encounter a domain shift if we have two domains that do not share distribution. As we do not know the underlying distribution, we have to find some measure on how much domain shift we encounter. In this perspective, we introduce the adaptation loss. Assume we train a classifier on a domain denoted by $\mathbb{P}$ and use this classifier to classify different data from the same domain, we will find the inner domain classification error $\xi(\mathbb{P})$. Assume we use the same classifier to classify data from another domain denoted by $\bar{\mathbb{P}}$, we will find the cross domain classification error $\xi(\mathbb{P}, \bar{\mathbb{P}})$. If the two distributions of $\mathbb{P}$ and $\bar{\mathbb{P}}$ differ, we expect a domain shift giving $\xi(\mathbb{P}, \bar{\mathbb{P}}) > \xi(\mathbb{P})$. The adaptation loss can be seen as a measure of how much domain shift we encounter. The adaptation loss of a classifier trained on $\mathbb{P}$ and tested on $\bar{\mathbb{P}}$, $a(\mathbb{P}, \bar{\mathbb{P}})$, is defined as:

$$a(\mathbb{P}, \bar{\mathbb{P}}) = \xi(\mathbb{P}, \bar{\mathbb{P}}) - \xi(\mathbb{P}). \tag{4-1}$$

Note that the adaptation loss is dependent on design choices in the SC pipeline such as, type of classifier and features used for classification. See chapter 5 for a more mathematical expression of the classification error and classifier.

4-2 Domain shift in SC

One of the biggest challenges in many SC applications, is the discrepancy between the source domain where the classifier is trained on, and the target domain of interest. Typically, we have large amounts of labeled data from different source domains, but only possess few, unlabeled data from the target domain, both originated from different underlying distributions. In this setting, cross domain classification for SC generally performs rather bad compared with inner domain training and testing. This so called domain-transfer problem can be intuitively explained due to different fashions of sentiment expression in different domains [46].

Let us quantify the assumption that SC is subjected to domain shift. First, we acquired 14 different datasets, see Table 4-1. We lower cased all words and removed punctuation. We choose to use all bigrams and unigrams as features when they occur more than 4 times in the corpus and at most in 40% of the documents. We weight the feature values by the TF-IDF weighting scheme, see section 3-3-3. As last step, we need to select a classifier. We used nine popular classifiers in default setting [47] and measured classification error and processing time, see Figure B-5. To make sure we measure the adaptation loss properly, we used the four best performing classifiers in terms of classification error: Ridge Regression (RR), Logistic Regression (LR), Multinomial NB (NB) and Linear Support Vector Machine (SVM). We measured the inner domain classification error, cross domain classification error and resulting adaptation loss, see Figure 4-1 and Figure 4-2, see Table B-1 for specific results on adaptation loss of each domain pair. From Figure 4-1 we see that the LR classifier has the lowest cross and inner domain classification error. Also, its computation time is more or less equal to the other evaluated classifiers. Therefore, we use a LR classifier in further experiments.

From Figure 4-2 we clearly see that we encounter a major domain shift in nearly every source-target domain pair. There is no evaluated scenario where a source domain is better to train on for a selected target domain than the target domain itself, although this could be possible when the source domain consists of more data and is similar to the actual target domain. The adaptation loss seems to follow a beta distribution. Also, we see that for a specific target domain, it is very beneficial to select the right source domain to train on. To state the relevance of the domain shift problem in SC:

1. Training on another domain than the target domain results in an average adaptation loss of 21.7 percent point,
2. Training on the best source domain, excluding the target domain itself, results in 6.7 percent point adaptation loss,
3. Training on the worst source domain gives an average adaptation loss of 39.6 percent point.

Notice that there is a lot to win if one is able to identify a 'good source domain'.

If we take a closer look at the adaptation loss for all the domain pairs, we see that there is a difference in the loss when using a domain as source domain, compared to using it as target. We see only little correlation between the two, see Figure 4-3. For clarification, Figure 4-3 shows the scatter plot of the average of each row versus the average over each column of Table B-1. The diagonal elements are discarded as these values represent the adaptation loss of inner domain classification, which is by definition zero.

4-3 Dealing with domain shift

In this chapter, we elaborate on two approaches to reduce the adaptation loss. First, by domain adaptation and second, by selecting a suitable source domain to train on. These two approaches can be combined to boost performance, this combined approach is however out of scope for this research.

Table 4-1: Used data sets for experiments.

Source	Topic	Number of docs	Doc. length	Std. doc length
Crunchr BV ⋆ [48]	Employee surveys	4474	46	65
University of Michigan [49]	Movies (1)	7085	11	7.2
CrowdFlower [50]	Books and movies	30602	14	6.9
Twitter [50]	Politics	10729	17	5.0
Twitter [50]	Self-driving cars	2698	16	5.2
Amazon [51]	Consumer electronics	1000	10	6.7
Yelp [51]	Venues	996	11	6.3
IMDB [51]	Movies (2)	746	20	67
Twitter [50]	Weather	502	14	6.3
Twitter [50]	Airlines	11541	19	6.6
Twitter [50]	Festival	2836	12	5.5
Twitter [50]	Sports game	8621	16	5.2
Twitter [50]	Apple computers	1642	16	6.0
Twitter [50]	Medicine	3244	13	6.0

⋆ A confidential data set that is not included in all simulations due to legal issues when results will be published.

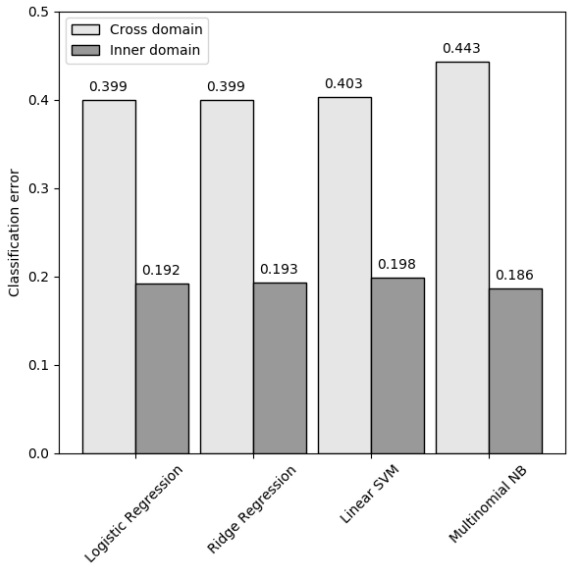


Figure 4-1: Inner and cross domain classification error of RR, LR, NB and SVM for used datasets.

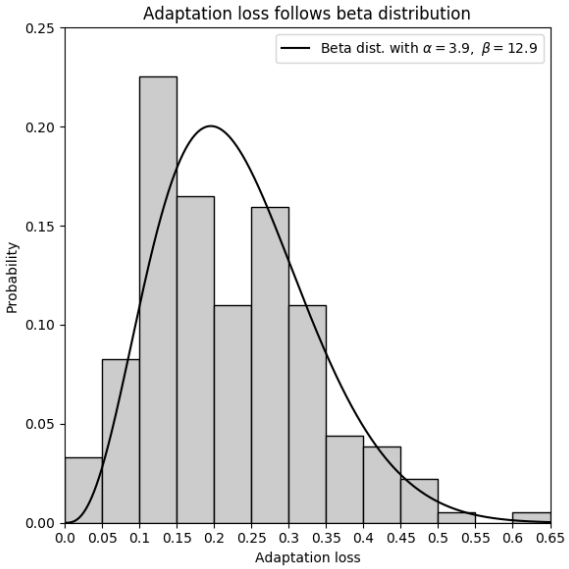


Figure 4-2: Distribution of adaptation loss averaged over RR, LR, NB and SVM classifier for used datasets.

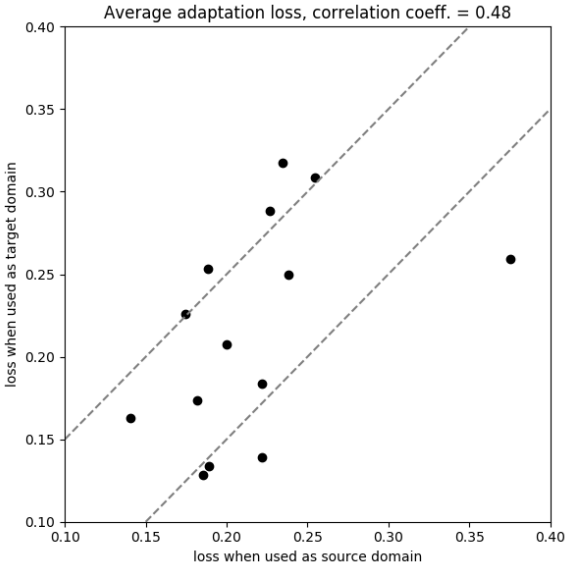


Figure 4-3: Scatter plot of adaptation loss averaged over RR, LR, NB and SVM classifier using a domain as source vs. using it as target.

4-3-1 Domain adaptation

In a naive training setting, we construct a classifier without looking at data of the target domain. However, in a domain shift situation, the source and the target distribution differ. Looking only at the source domain data might result in a bad performing classifier. There are machine learning techniques that look at both source and target data in order to increase performance in a domain shift training setting. In a domain shift setting, when we use unlabeled data of the target domain in the construction of the feature space or classifier, we make use of domain adaptation or transfer learning techniques. Three main categories of domain adaptation techniques can be distinguished: importance sampling, feature space transformations and self learning or iterative transfer learning.

When we make use of importance sampling, we assign (importance) weights to the source objects, documents in our case study, in such a way that the distribution of source and target data are more similar. In order to do so, a distance function has to be defined. The weights on the source sample ought to minimize the distance function. A popular approach is Kernel Mean Matching. This method finds importance weights by minimizing the MMD, see section 5-2-1 for a mathematical description of the MMD [52].

The second class of techniques modifies the feature space to a new, often lower dimensional, feature space in which classification will be performed. A well known approach is Structural Correspondence Learning (SCL) [53]. With SCL one tries to identify correspondences among features from the source and features from the target domain by modeling their correlations with pivot features. Pivot features are features that occur frequent or behave the same in both source and target domain. Non-pivot features that are correlated with a pivot feature will be treated as if they hold similar sentiment. This allows us to, for example, use words of the target domain that do not occur in the source domain for classification. Another technique is called Transfer Component Analysis (TCA) [54]. TCA tries to construct a transformation to a lower dimensional embedded space in which the source and target distribution are similar. Similarity is, again, measured with the MMD. One of the relatively new approaches is Subspace Alignment. To perform Subspace Alignment [55], one performs a Principle Component Analysis to select eigenvactors. The eigenvectors are used to span a subspace. Next, the source data in the subspace is transformed with an affine transformation. The transformation is constructed in such a way that the transformed source and original target data are similar in the subspace spanned by the eigenvectors. Similarity is measured with the Frobenius norm, and is minimized.

A last category of domain adaptation techniques is self learning transfer learning. These methods use iteration to improve the classifier. In the first step, a classifier is trained on the source data. It is then applied to the target data to establish predictions for their labels. These predicted labels of the target domain are then used to update the classifier. These last two steps are repeated. A transformation matrix is constructed through iteration, for example based on maximum entropy [56].

4-3-2 Source domain selection

Apart from performing domain adaptation, if multiple labeled source domains are available, one could choose to train on all or on a subset of all possible candidate source domains. In

many cases, it appears natural which source domain to choose. Let us assume that we would like to classify the sentiment on tweets about Dutch right wing politician Geert Wilders. If we would have lots of labeled tweets on Wilders, we would simply train a classifier on this labeled data. Often, we do not have labeled data from the target domain to train on and we have to select another domain to train a classifier. Let us again assume that we want to classify the tweets on right wing politician Wilders. Now, we have access to multiple labeled data sets with infinite labeled objects. We have a data set concerning tweets about the Dutch Royal Airlines, the new iPhone and the Dutch prime minister. It feels natural to assume that we should train on the data set with tweets about the prime minister as this lays close in topic and source audience to the target domain. But in some cases, the choice is little harder. In a new scenario, again, we would like to classify tweets on Geert Wilders. We got hold of labeled tweets on the Dutch prime minister, left wing politician Marianne Thieme and Dutch tweets about right wing US politician Mike Pence. On which one should we train? Or if we would like to classify reviews on the new BMW sports car, should we train on labeled movie reviews, reviews of restaurants or reviews on consumer electronics? Or should we train on all of them? The process of selecting one or multiple source domains from a set of candidate source domains is referred to as source domain selection.

Source domain selection has been researched and used before in the light of applications in several fields. For example, to improve brain-computer interface calibration, supervised source domain selection based on distance measured as distance between the class average vectors has been successfully used [57]. For a more open search space for a suitable source domain, techniques have been researched to find a source domain in open online information sources such as Wikipedia [58]. However both approaches are supervised as they need class labels of the target domain. In many cases, we do not have any class labels of the target domain.

In the field of sentiment analysis, Blitzer et al. show a very good correlation between the $\mathcal{A}$-distance and adaptation loss [58]. However, the adaptation loss is measured using the target domain inner domain classification error, $\xi(\bar{\mathbb{P}})$. In many cases, we do not have any labeled information on the target domain, making source domain selection based on the predicted adaptation loss impossible in the way it is proposed by Blitzer. Also, remarkably, for the proof of correlation, half of the domains is not shown in the figure which may rise some eyebrows. The $\mathcal{A}(\mathbb{P}, \bar{\mathbb{P}})$-distance between $\mathbb{P}$ and $\bar{\mathbb{P}}$ is defined as

$$\mathcal{A}(\mathbb{P}, \bar{\mathbb{P}}) = 2 \sup_{A \in \mathcal{A}} \left| P(x \in A \mid x \sim \mathbb{P}_x) - P(x \in A \mid x \sim \bar{\mathbb{P}}_x) \right|$$

where $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$ denote the marginal distribution of the source and target domain respectively, i.e. the distribution of unlabeled data. The computation of this optimization problem is however not convex. A proxy is used to approximate the $\mathcal{A}$-distance. Blizter et al. use modified Huber loss as proxy. Notice that this optimization problem actually is similar to a classification process. In this optimization, a classifier is fitted by minimizing the modified Huber loss, in order to classify each document to its source domain. When this classifier would be perfect, it is able to perfectly identify which document comes from which domain. When the classifier error is zero, the domains are completely separable and are assumed to be very different. Other proxies for the $\mathcal{A}$-distance can be used. In recent research in orthophoto classification, the $\mathcal{A}$-distance is, for example, approximated by the MMD in stead of the Huber

loss [58]. Apart from the $\mathcal{A}$-distance, there have been studied some other approaches in the field of SC such as selecting individual source documents based on cross entropy [59].

For 13 acquired data sets, we tested the performance of using the $\mathcal{A}$-distance to select the one best source domain from 12 candidates. We used the data sets described in 4-1 except the confidential Crunchr data set. The results were poor, improving the cross domain classification error only by .5 percent point compared to random domain selection.

In the next chapters, we will propose a new method of selecting a suitable source domain from a set of candidates without the need for any labeled data from a target domain of interest.

Chapter 5

CMEK source domain selection

In this chapter we will propose a new, unseen method to select one or multiple source domains. We will start by properly defining the problem and the challenges it brings for selecting the best source domain, not specified to the field of SC. Then, we will propose a method to address this challenge by using four distance measures. Using a linear combination of the four measures, a constant and the inner domain classification error of the source domains for source domain selection is introduced as the CMEK source domain selection model. We state some important limitations of our approach and how we will evaluate its performance.

5-1 Problem definition

With a document represented by a random variable X and its sentiment label by random variable Y, each domain is uniquely characterized by its joint probability density function supported on $\mathbb{X} \times \mathbb{Y}$. Let us then define two underlying distributions to the evaluated source and target data on $\mathbb{X} \times \mathbb{Y}$, denoted by $\mathbb{P}$ and $\bar{\mathbb{P}}$ respectively. Their marginals on $\mathbb{X}$, are denoted as $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$ respectively. In machine learning, a realization of $\mathbb{P}$ is referred to as labeled data whereas realizations from $\mathbb{P}_x$ are referred to as unlabeled data.

We define an hypothesis function as a mapping from $\mathbb{X}$ to $\mathbb{Y}$, $h : \mathbb{X} \to \mathbb{Y}$. In the context of classification problems in the machine learning literature, the set $\mathbb{Y}$ is often discrete, often binary, and the hypothesis h is often referred to as a classifier as it is able to assign a class label in $\mathbb{Y}$ to each data point in $\mathbb{X}$. The true hypothesis function $h^\star_{\mathbb{P}}$ for a domain characterized by $\mathbb{P}$, is defined as

$$h^\star_{\mathbb{P}} := \arg\min_h P\big(h(x) \neq y \mid (x, y) \sim \mathbb{P}\big). \tag{5-1}$$

We then define the inner domain classification error as the probability of misclassification in the same domain as the true hypothesis function is constructed in. And the cross domain

classification error is defined as the probability of misclassification in the target domain of the true hypothesis function of the source domain

$$\begin{aligned} \xi(\mathbb{P}) &:= P\big(h^{\star}_{\mathbb{P}}(x) \neq y \mid (x,y) \sim \mathbb{P}\big). \\ \xi(\mathbb{P},\bar{\mathbb{P}}) &:= P\big(h^{\star}_{\mathbb{P}}(x) \neq y \mid (x,y) \sim \bar{\mathbb{P}}\big). \end{aligned} \tag{5-2}$$

Note that due to the stochastic nature of sentiment expression, $\xi(\mathbb{P})$ and $\xi(\mathbb{P},\bar{\mathbb{P}})$ are not likely to be zero, even if we would have infinite data to train on. In addition, due to the difference in source and target distribution, $\xi(\mathbb{P},\bar{\mathbb{P}})$ is likely to be higher than $\xi(\mathbb{P})$ [58]. The difference between the cross and inner domain classification error is referred to as the adaptation loss.

The goal in many classification tasks is to minimize the cross domain classification error (5-2) for a specific target domain while we have access to a finite set of labeled candidate source domains. An appropriate choice of the source domain in this context is the main goal of this study, leading to the following question:

What source domain minimizes the cross domain classification error for a given target domain?

With the choice of source domain restricted to a domain characterized by a probability density function in the candidate set $\mathcal{P}$, our main goal can be formally described through the optimization program

$$\mathbb{P}^{\star} := \arg\min_{\mathbb{P}\in\mathcal{P}} \xi(\mathbb{P},\bar{\mathbb{P}}). \tag{5-3}$$

where $\xi(\mathbb{P},\bar{\mathbb{P}})$ is the cross domain classification error introduced in (5-2). In other words, when we posses a finite number of labelled data sets, each originated from a unique source domain, on which set should we train our classifier in order to minimize classification error on data from a specific target domain?

The challenge concerning the objective (5-3), is that for a target domain of interest we typically only have unlabeled data, therefore we cannot calculate the cross domain classification error. This means we only know the marginal distribution $\bar{\mathbb{P}}_x$ of our target domain instead of $\bar{\mathbb{P}}$, making it impossible to explicitly calculate $\xi(\mathbb{P},\bar{\mathbb{P}})$. In order to find the best source domain characterized by $\mathbb{P}^{\star} \in \mathcal{P}$ for our target domain with distribution $\bar{\mathbb{P}}$, we ought to predict $\xi(\mathbb{P},\bar{\mathbb{P}})$ for all $\mathbb{P} \in \mathcal{P}$. For this prediction we have full distributional information of the source domains, $\mathbb{P}$, and marginal information on the distribution of the target domain, $\bar{\mathbb{P}}_x$, available.

5-2 CMEK model

In this section, we propose a model, called CMEK source domain selection, to deal with the challenge stated in section 5-1. The model measures statistical distances between the

marginal distributions of candidate source domains and the target domain and uses the inner domain classification error of the candidate source domains. The candidate source domain with the lowest distance is hypothesized to have the lowest cross domain classification error and is selected to train a classifier. This section will describe the CMEK model, the distance measures it uses and how we will evaluate the performance of the model.

We consider a set $\mathcal{D}$ containing a family of distance functions d: $\mathbb{P} \times \bar{\mathbb{P}}_x \rightarrow \mathbb{R}_+$. We now hypothesize that given an available set of source domain distributions $\mathcal{P}$ and a target domain distribution denoted by $\bar{\mathbb{P}}$ with the marginal $\bar{\mathbb{P}}_x$, there exists a $\hat{d} \in \mathcal{D}$ that can reliably predict the cross domain classification accuracy $\xi(\mathbb{P}, \bar{\mathbb{P}})$, that is,

$$\xi(\mathbb{P}, \bar{\mathbb{P}}) \overset{\text{hyp}}{\approx} \hat{d}(\mathbb{P}, \bar{\mathbb{P}}_x). \tag{5-4}$$

More specific, we hypothesize that the cross domain classification error can be predicted by looking at the difference in marginal distribution functions $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$ and $\xi(\mathbb{P})$. With this prediction, the best candidate source domain can be selected for training. We formalize this optimal choice of the measure by considering the optimization problem

$$\hat{d} := \arg\min_{d \in \mathcal{D}} \sum_{\mathbb{P} \in \bar{\mathcal{P}}} \left| \xi(\mathbb{P}, \bar{\mathbb{P}}) - d(\mathbb{P}, \mathbb{P}_x) \right|. \tag{5-5}$$

To construct the family $\mathcal{D}$ of candidate measures, we use a vector s with K known statistical distance measures as elements, s_i: $\mathbb{P}_x \times \bar{\mathbb{P}}_x \rightarrow \mathbb{R}_+$ for $i \in \{1, \cdots, K\}$. The family $\mathcal{D}$ consists of linear combinations of these measures s_i with corresponding weight coefficients $\beta_i \in \mathbb{R}_+$, with β_i being elements of the weight vector β. We augment the linear combination with a constant. Note that the distance measures are functions supported on the marginal distributions of source and target domain. Examples of these statistical distance metrics are the Chi squared distance (Chi2), Maximum Mean Discrepancy (MMD), Earth Mover's Distance (EMD) and Kullback-Leibler Divergence (KLD). When a source domain has poor inner domain classification performance, we expect it to be less suitable as source domain for cross domain classification. Therefore, we calculate the inner domain classification error for all candidate source domains and add this to our vector s. With our choice for $\mathcal{D}$, the optimization problem (5-5) can be reduced to

$$\begin{aligned} \hat{\beta} &:= \arg\min_{\beta \geq 0} \sum_{\mathbb{P} \in \mathcal{P}} \left| \xi(\mathbb{P}, \bar{\mathbb{P}}) - \beta s(\mathbb{P}, \bar{\mathbb{P}}_x) \right|, \\ \hat{d} &:= \hat{\beta} s(\mathbb{P}, \bar{\mathbb{P}}_x) \end{aligned} \tag{5-6}$$

where $\hat{\beta}$ denotes the optimal weight vector. Note that program (5-6) constructs the optimal unsupervised predictor by using the full distribution of the target data, which is presumed to be unknown. In a practical setting with a finite number of elements in $\mathcal{P}$, we deal with this problem by training only on the domains in $\mathcal{P}$. In each run, one of the elements of $\mathcal{P}$ is extracted from the set and used as proxy $\bar{\mathbb{P}}$, at the end of the run, this element, denoted as $\tilde{\mathbb{P}}$ and its marginal $\tilde{\mathbb{P}}_x$, is placed back in $\mathcal{P}$. In every run, another element is extracted and placed back for training. This training program is described as

$$\hat{\beta} := \arg\min_{\beta \geq 0} \sum_{\mathbb{P} \in \mathcal{P}, \mathbb{P} \neq \tilde{\mathbb{P}}} \sum_{\tilde{\mathbb{P}} \in \mathcal{P}} \left| \xi(\mathbb{P}, \tilde{\mathbb{P}}) - \beta s(\mathbb{P}, \tilde{\mathbb{P}}_x) \right|.$$

We hypothesize that with a sufficient amount of source domain distributions in $\mathcal{P}$, the constructed predictor for the cross domain classification error, based on information from $\mathcal{P}$, is also reasonably accurate in predicting the cross domain classification error for a target domain not included in $\mathcal{P}$. This allows us to select the best source domain to train on for this unseen target domain.

Now, in view of our hypothesis (5-4), with the predictor $\hat{d}$ we are able to approximate $\xi(\mathbb{P}, \bar{\mathbb{P}})$ with only marginal information of the target domain $\bar{\mathbb{P}}_x$, which allows us to deal with the challenge through an approximation. This enables us to, given a target domain with distribution $\bar{\mathbb{P}}$, select the source domain, characterized by $\hat{\mathbb{P}}$ from a set of available source domains that is predicted to minimize the cross domain classification error $\xi(\mathbb{P}, \bar{\mathbb{P}})$ as formalized through the optimization program

$$\hat{\mathbb{P}} := \arg\min_{\mathbb{P} \in \mathcal{P}} \hat{d}(\mathbb{P}, \bar{\mathbb{P}}_x)$$

which is the goal of this study.

5-2-1 Measures

For our SC case study, we hypothesize that the way people express sentiment can be captured in how often certain words and word combinations are used. The underlying joint distributions $\mathbb{P}$ differs among different domains. To accurately classify sentiment for a target domain, we therefore ought to find a domain with similar underlying distribution function. But how do we measure similarity between distributions?

When we talk about statistical similarity or distance measures, we can define two different approaches of measuring distance. First, we can compare statistical meta data such as the average word frequency. The Chi2 distance is based on comparing this average word frequency between two evaluated corpora. Other meta approaches are, for example, based on comparing rank of word frequency. These approaches are in general computational cheap, but not very robust. Another approach is to compare the distribution of word frequency. These approaches are referred to as Integral Probability Metrics (IPMs). Where the previously described metrics only compare meta data on the word frequency distributions, IPMs compare the actual distributions. IPMs use a class $\mathcal{F}$ of functions $f : \mathbb{X} \to \mathbb{R}$ to calculate the distance s_i between $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$ as

$$s_i(\mathbb{P}_x, \bar{\mathbb{P}}_x) = \sup_{f \in \mathcal{F}} \left| \mathbb{E}^{\mathbb{P}_x}\left[f(X)\right] - \mathbb{E}^{\bar{\mathbb{P}}_x}\left[f(X)\right] \right|, \tag{5-7}$$

the question arises which class $\mathcal{F}$ to choose [60]. If we would choose the 1-Lipschitz function for $\mathcal{F}$, we get the popular and straightforward EMD metric [61]. This metric is straightforward as it measures how much distribution mass has to be moved to transform one distribution in another. The mass is weighted proportionally to the distance it has to travel. If we would

choose $\mathcal{F}$ in (5-7) to be a class of functions in the unit ball of a reproducing kernel Hilbert space, we would have the MMD metric [62]. This approach finds a well behaved, smooth, function which is high on the points drawn from $\mathbb{P}_x$ and low on $\bar{\mathbb{P}}_x$ or vice verse. We use as our test statistic the difference between the mean function values on the two samples according to (5-7) [63]. Another approach is to choose a logarithmic function for f, giving the KLD with the interesting asymmetry property $s_i(\mathbb{P}_x, \bar{\mathbb{P}}_x) \neq s_i(\bar{\mathbb{P}}_x, \mathbb{P}_x)$. As difference in distribution becomes larger, KLD gives decreasingly increasing distance.

Since we hypothesize that the way people express sentiment can be captured in how often certain words and word combinations are used, but not know in what relationship, we propose a broad spectrum of possibilities to optimize over. We propose to evaluate a meta approach, Chi2 distance, and three IPM approaches that use different comparison functions f: MMD, EMD and KLD. Therefore we have four measures, a constant and the inner domain classification error of the source domain that span the distance vector space $\mathcal{D}$. In the next paragraphs we will briefly elaborate on how we use these distance measures in a discretized fashion. Selection based on the linear combination of these four metrics, the constant and $\xi(\mathbb{P})$ will be referred to as the CMEK selection model.

Chi2

As first distance measure, we make use of the test statistic of Pearson's χ^2 test [64]. Let us have two unlabeled corpora, c_x and $\bar{c}_x$, containing the documents of the source and target domain respectively. Then, let $p(w)$ be the probability of a word in c_x being w, and $\bar{p}(w)$ indicating the probability of a word being w in $\bar{c}_x$. We calculate the value of $Chi2(c_x, \bar{c}_x)$ as the sum of Pearson's test metric over the N most occurring words, $w_1, w_2, \ldots, w_N$,

$$Chi2(c_x, \bar{c}_x) := \sum_{i=1}^{N} \frac{\left(\left(p(w_i) - \bar{p}(w_i)\right)\right)^2}{p(w_i) + \bar{p}(w_i)}.$$

MMD

Our second distance measure is the Maximum Mean Discrepancy (MMD) defined as the largest difference in expectations over functions in the unit ball of a reproducing kernel Hilbert space [62]. Let a unit ball $\mathcal{F}$ in a universal reproducing kernel Hilbert space $\mathcal{H}$ be a class of functions $f : \mathbb{X} \to \mathbb{R}$ and $\mathbb{E}^{\mathbb{P}}[X]$ be the expected value of random variable X distributed according distribution $\mathbb{P}$. Then the MMD is defined as

$$MMD(\mathbb{P}_x, \bar{\mathbb{P}}_x) := \sup_{f:||f||_{\mathcal{H}} \leq 1} \left| \mathbb{E}^{\mathbb{P}_x}\left[f(X)\right] - \mathbb{E}^{\bar{\mathbb{P}}_x}\left[f(X)\right] \right|. \tag{5-8}$$

EMD

The Earth Mover's Distance (EMD), similar to the Wasserstein metric [65], is a broadly used metric to compare distributions. The metric measures how much probability mass has

to be moved, and how far it has to be moved, to transform two distributions into each other. Imagine two discrete distribution functions $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$ with cluster representatives b and $\bar{b}$ both with m clusters and their corresponding probabilities p and $\bar{p}$ adding both up to one. This can be visualized by a histogram with m bins, centered on b, with heights p, and a similar representation for the target domain data. Then, we define a distance matrix $D[=d_{i,j}] \in \mathbb{R}^{m \times m}$, indicating the distances between the two cluster representatives b and $\bar{b}$. At last we define the flow matrix $F = [f_{i,j}] \in \mathbb{R}^{m \times m}$ as the flow of probability mass that has to be moved to transform the two distribution into each other. The EMD value between the two distributions is the minimized weighted, by D, flow needed for transformation,

$$\begin{aligned} EMD(\mathbb{P}_x, \bar{\mathbb{P}}_x) := \quad & \min_F \sum_{i=1}^{m} \sum_{j=1}^{m} d_{i,j} f_{i,j} \\ \text{subject to} \quad & f_{i,j} \geq 0 \\ & \sum_{i=1}^{m} f_{i,j} = p_i \\ & \sum_{j=1}^{m} f_{i,j} = \bar{p}_j. \end{aligned}$$

KLD

As last measure, we use the Kullback-Leibler Divergence (KLD) [66], also known as relative entropy [67]. For two discrete probability functions $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$, the natural logarithm of the ratio of their two probability values p and $\bar{p}$ respectively are basis for the KLD value. With cluster representatives $b = \bar{b}$ and m clusters, the KLD value is defined as

$$KLD(\mathbb{P}_x, \bar{\mathbb{P}}_x) := \sum_{i=1}^{m} p_i \log\left(\frac{p_i}{\bar{p}_i}\right).$$

Notice that this metric is asymmetric which might come in handy as the cross domain classification error is asymmetric as well.

5-2-2 Performance evaluation

We have constructed a model that predicts cross domain classification error between a candidate source domain and a target domain with a linear combination of statistical distance measures based on the marginal distributions $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$, a constant and $\xi(\mathbb{P})$. Now we are interested in how well this model is able to identify the best candidate source domain, i.e. the one that gives the lowest cross domain classification error $\xi(\mathbb{P}, \bar{\mathbb{P}})$. For n_c acquired data sets, let us use one corpus as target domain, $\bar{\mathbb{P}}$, and the others as candidate source domains with the set annotation $\mathcal{P}'$. We can repeat the experiment n_c times by using each time a different target domain and average the results.

We first let the model select the predicted best source domain $\hat{\mathbb{P}} \in \mathcal{P}'$ and use this domain for training. To evaluate performance we define the relative cross domain classification error

$\xi_{relative}(\hat{\mathbb{P}}, \bar{\mathbb{P}})$ as

$$\xi_{relative}(\bar{\mathbb{P}}) = \xi(\hat{\mathbb{P}}, \bar{\mathbb{P}}) - \xi(\mathbb{P}^\star, \bar{\mathbb{P}}) \tag{5-9}$$

where $\mathbb{P}^\star$ is the domain in $\mathcal{P}'$ of which we know it has the lowest cross domain classification error, the true best domain. When the model selects the best source domain, the relative error will therefore be zero. We perform the same analysis when using each measure individually as predictor. We calculate the average relative error but also look at the distribution of this relative error to see how often our models selects one of the best candidate source domains and how often it fails and selects a poor candidate source domain.

Next, we let our proposed CMEK model select the best n domains from $\mathcal{P}'$ and compare $\xi(\bigcup_{i=1}^{n} \hat{\mathbb{P}}_i, \bar{\mathbb{P}})$ with training on all domains in $\mathcal{P}'$. We perform the same analysis when using the Chi2 distance and EMD measure individually. All results will be tested on significance using a paired t-test over the results of 13 runs, we will use a significance level of $p = 0.05$. The paired t-test is strongly dependent on the assumption that the pairwise differences are normally distributed. For every comparison between results, we evaluated the normality assumption. We report only on significant results for which the normality assumption is confirmed.

5-3 Limitations

In solving the problem as defined in section 5-1 with our proposed method, we encounter limitations of various kinds. We first discuss a fundamental statistical limitation that is present in almost all practical machine learning problems. Then we discuss a limitation of computational nature and how to deal with this. In the last paragraph we elaborate on limitations that are more specific to the case study of SC: numerically representing text and the limited hypothesis space. We end with briefly explaining the limitations of our proposed CMEK model.

5-3-1 Statistical limitations

In the first place, and inherent to all classification problems, we are unable to retrieve the true distribution over the support set $\mathbb{X} \times \mathbb{Y}$ in practical settings. Instead, we need to infer this distribution through only a finite number of observations. In this case a natural approximation for the distribution is the empirical distribution supported on these observations, which are the available documents in our SC case study. We then encounter an inevitable approximation

$$\mathbb{P} \approx \frac{1}{n_s} \sum_{i}^{n_s} \delta_{\{x_i, y_i\}}$$

where n_s denotes the number of documents in the domain sample. Due to the approximation of the true distribution, the hypothesis function (5-1) will be suboptimal, resulting in a higher classification error. To evaluate this error, the available data originated from one domain is split into a train and test set. The train set is used to construct $h^\star_{\mathbb{P}}$ as defined in (5-1). The

test set is used to calculate the inner domain classification error. When using sufficient data in the test set, a large difference in apparent and true error indicates that the observations used for training are a poor support for approximating the true distribution since it means that the empirical distributions from the train and test set are not much alike. When this is the case, we are over fitting; the hypothesis function works for the data it is constructed on, but can not be properly generalized to perform well on new data from the same underlying distribution. To reduce over fitting, we ought to use a sufficient number of training objects and a low but not too low number of features.

5-3-2 Computational limitations

Another limitation is that the optimization in (5-1) uses the indicator function as loss function, which is non-convex. For computational reasons, we replace this loss function with a convex counterpart. For the true hypothesis function $h^\star_{\mathbb{P}}$ in (5-1), a common convex loss function $\ell(x, y)$ to minimize, is the quadratic error of the prediction, popular as the method of least squares

$$h^\star_{\mathbb{P}} = \arg\min_h \sum_{i=1}^{n_s} \ell(x_i, y_i) \qquad \ell(x, y) := \big(h(x) - y\big)^2, \tag{5-10}$$

and the cross domain classification error $\xi(\mathbb{P}, \bar{\mathbb{P}})$ will be defined by empirical distributions of the target and the hypothesis function from (5-10),

$$\xi(\mathbb{P}, \bar{\mathbb{P}}) := \frac{1}{n_s} \sum_{i}^{n_s} \mathbb{1}_{\left\{h^\star_{\mathbb{P}}(x_i) \neq y_i \;\mid\; (x,y)\sim\bar{\mathbb{P}}\right\}}. \tag{5-11}$$

5-3-3 Further discussion on SC related limitations

There are limitations concerned with performing SC with a nature that may also be relevant to other fields of application. Choosing a word representation model is a field of study by itself and should be considered carefully here. In the previous section we assumed a true hypothesis function $h^\star$ mapping each document represented in $\mathbb{X}$ to a label in $\mathbb{Y}$. For computational purpose, we need to numerically express documents in the $\mathbb{X}$ space. Let us call W the set of all words, $W = \{w_1, w_2, \dots w_{n_w}\}$ where n_w is the number of unique words. Then, documents in $\mathbb{X}$ are combinations of these words, indicating that $\mathbb{X}$ can be viewed as an element in the power set of W denoted by $\wp(W)$. Note that repetition in the subset is possible. For computational feasibility, we choose a subset $\mathcal{F} \subset \wp(W)$ to represent our documents. The subset $\mathcal{F}$ is called the feature set whose cardinality is denoted by $|\mathcal{F}|$. The features selection process depends on what representation model is at hand. The representation model $\mathfrak{F}$ maps $\mathbb{X}$ to $\mathbb{R}^{|\mathcal{F}|}$, i.e. $\mathfrak{F} : \mathbb{X} \to \mathbb{R}^{|\mathcal{F}|}$. The computational need for a representation model $\mathfrak{F}$ to select a subset of $\mathbb{X}$ for representation, limits the classification model since information has to be discarded, see Figure 5-1.

Furthermore, in the task of classification we need to define a search space $\mathcal{H}$ for our hypothesis function as it is computational impossible to search through all possible hypothesis functions

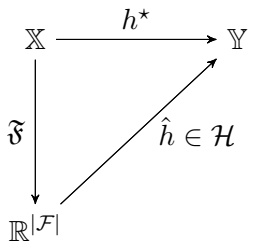


Figure 5-1: Representation and classification structure

as proposed in (5-1). SC tasks are characterized by the fact that the representation space is sparse, i.e. the ratio of number of training documents to $|\mathcal{F}|$ is significantly low. In this situation, computational feasibility calls for a tractable subset of hypothesis functions $\mathcal{H}$ such as all linear functions. The true hypothesis function is not likely to be in our hypothesis space, $h^\star \notin \mathcal{H}$. We define the optimal hypothesis function constructed in the domain $\mathbb{P}$ as the hypothesis $\hat{h}$ within our limited hypothesis space $\mathcal{H}$, which is the collection of all linear mappings from $\mathbb{R}^{|\mathcal{F}|}$ to $\mathbb{R}$, that minimizes the chosen loss function (5-10). The cross domain classification error from (5-11) is calculated with the optimal hypothesis function $\hat{h}_{\mathbb{P}}$.

Hereafter, we use $\mathbb{P}$ to annotate the empirical distribution instead of the true continuous distribution. Furthermore, the cross domain classification error defined as $\xi(\mathbb{P}, \bar{\mathbb{P}})$ in (5-2) is calculated with the optimal hypothesis function in $\mathcal{H}$ that minimizes the empirical loss in (5-10).

5-3-4 CMEK selection model limitations

Our model assumes that a good source domain, with low cross domain classification error, can be identified by the distributions $\mathbb{P}$ and $\bar{\mathbb{P}}_x$. However, where in reality sentiment is determined by joint distribution of the features, the measure we use only measure distance on the individual distributions of the features. This means we try to find similarity of the joint distributions from marginal distributions of individual features. The same holds for classification that is based on marginal data of individual features whereas sentiment in real life is determined by combinations of features. Since we use marginal data for calculating the cross domain classification error as well as the prediction of this error, this approach seems justified, but it creates a model separated from the true fashion of expressing sentiment.

In addition, we do not know in what way we should compare distributions, only that when distributions are the same, we expect no adaptation loss. It might very well be that the linear combination of our chosen measures does not include the true distance function that distinguishes domains from each other.

Chapter 6

Experimental setup

In this Chapter we briefly elaborate on the data sets we used for our experiments and the design choices we made concerning the SC pipeline and used distance measures.

6-1 Data sets

We acquired 13 data sets, corpora, each consisting of a different number of documents ranging from 502 to 30602 documents with an average of 6326 documents, see Table 4-1. The average length of each document among the corpora varies between 10 words and 20 words with an average of 15 words. We used a corpus of online reviews on movies, venues and consumer products [51], short opinions about movies [49] and nine sets of tweets with various topics [50]. The acquired data sets do not have any major class imbalances.

6-2 Design choices

Considering the SC pipeline design choices, in our experiments we only remove punctuation and lower case all words as prepossessing. For classification in our experiments, we use a bag-of-words approach with unigrams and bigrams and use all features that occur more than 4 times in the training corpus and at maximum in 40% of the training documents. These choices are supported on outcomes of experiments, see Appendix B Figures B-1 and B-2 We weight the features according to the TF-IDF weighting scheme. We deliberately do not use word embeddings since the lower dimensional projection needs too much textual domain data to construct. Using data from other domains will distort results as the representation becomes dependent on other domains than the source and target domain that are under performance evaluation. We use LR for its good performance. We use binary classification, $y \in \{0, 1\}$, giving the following loss function for our classifier

$$\ell(x, y)_{LR} = \sum_{i=1}^{n_d} \log\left(1 + e^{-(2y_i - 1)x_i\beta + \beta_0}\right) + \alpha\frac{1}{2}\|\beta\|^2$$

where w and w_0 represent the optimizers, α represents the regularization parameter and the hypothesis function is defined as

$$h(x) = \begin{cases} 1 \text{ if } \beta x + \beta_0 > 0 \\ 0 \text{ else.} \end{cases}$$

For α, we use the default classifier settings of our used toolbox [47].

For the Chi2 distance, we use the number of occurrences of a word divided by the occurrences of the top $N = 1000$ words for calculating the probabilities. The $N = 1000$ most occurring words are used to compare means. For the MMD we use a discrete version of (5-8) [68] where X represents the feature value matrix for the $N = 1000$ most occurring features in the combined set of the two evaluated corpora. We use corpora of the same number of objects, i.e. documents, as input by using a subset of the largest corpus of the two corpora used. For computational convenience we use a maximum of 5000 documents. For EMD, we compare the discrete normalized distributions of the feature count in one document and use the sum of the EMD over the $N = 1000$ most occurring words with available EMD code [69] [70] [71]. Note that the distribution is highly dependent on the length of a document. Therefore, we match document lengths among the two domains. We use the same cluster representatives for both domains, $b = \bar{b} = \{0, 1, \ldots, m\}$ and a distance matrix $D = [d_{i,j}]$ defined by $d_{i,j} = |i - j|$. For the KLD we again use the discrete normalized distributions of the feature count in one document of $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$ and calculate the KLD as the sum over the $N = 1000$ most occurring features.

We do not compare the distribution of all features for two reasons. In the first place, the distance calculations will take long when using all features. And second, and more important, distributions of rarely occurring features are less reliable.

For the inner domain classification error that is used as predictor in the CMEK model, we use the same logistic regression as for the cross domain classification and use 10-fold cross validation to assess the inner domain classification error.

Chapter 7

Results

In this chapter we will objectively present the results of our experiments according to our performance evaluation described in subsection 5-2-2 without any interpretation. Interpretation of the results will be described in the next chapters.

Figure 3 shows the empirical distribution of the relative error as defined in (5-9) when selecting one source domain. The relative error distributions when randomly selecting a source domain and when using the CMEK source domain selection model are shown.

Table 7-1 shows the probability of selecting the true best domain, the probability of selecting one of the five worst domains in terms of the cross domain classification error, and the average cross domain classification error, averaged over the 13 different target domains. We listed the results when individual measures are used for selection, when the linear combination is used (CMEK) and when a source domain is selected at random. The optimal cross domain classification error is listed which can be seen as lower bound of error. The CMEK model uses the optimized weights

$$\hat{\beta} = [0.04,\ 1.19,\ 0.12,\ 0.05,\ 0.66,\ 0.25] \tag{7-1}$$

for respectively, Chi2, MMD, EMD, KLD, the source domain inner domain classification error and the constant 1.

Figure 7-2 shows the cross domain classification error when we let our selection model select n source domains. Results are shown when using CMEK, Chi2, EMD and random selection. Note that the CMEK model is constructed by optimizing the predicted cross domain classification error when training the classifier on single source domains, not training on multiple. The results of using these models for source domain selection are compared with training on all candidate source domains. We choose to present the Chi2 and EMD measures as they show the most interesting behavior over using n domains to train on. The MMD measure has similar behavior as the CMEK model and KLD selection behaves similar to EMD selection.

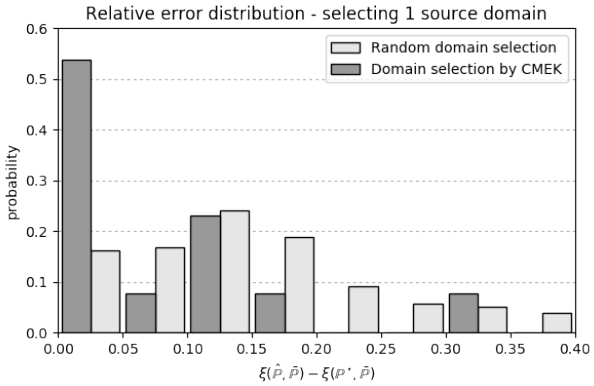


Figure 7-1: Performance of CMEK source domain selection model compared to random source domain selection.

Table 7-1: Performance of CMEK, individual measures, random and optimal selection

Selection method	**Probability Best possible domain selected**	**Probability One of 5 worst domains selected**	**Average** $\xi(\hat{\mathbb{P}}, \bar{\mathbb{P}})$
Optimal	1	0	.252
EMD	.308	.000	.309
KLD	.308	.077	.324
CMEK	.385	.154	.330
MMD	.077	.154	.350
Chi2	.154	.308	.369
Random	.083	.417	.403

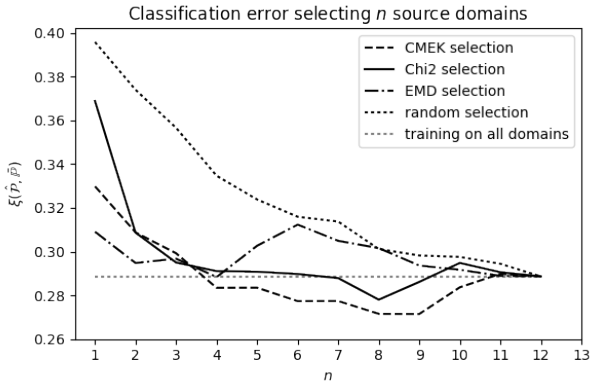


Figure 7-2: Performance of CMEK model, Chi2, EMD and random selection for different number of source domains to be selected, compared with training on all source domains.

7-1 Significance of the results

In determining the significance of the results, we cannot simply calculate the deviation of the results. The optimal, random or selection model results are dependent on the target domain; the deviation over the 13 runs, i.e. 13 different targets domains, tells us nothing about significance. Instead, to compare two methods with each other, we look at the distribution of the pair wise difference by performing a paired t-test.

The paired t-test subtracts the results of two models from each other, we denote the difference between the N results of model M_j and M_k by δ,

$$\delta(M_j, M_k) = \{\delta_1, \delta_2, \dots, \delta_N\} \qquad \delta_i = [\xi(\hat{\mathbb{P}} \in \mathcal{P}', \bar{\mathbb{P}}_i)|M_j] - [\xi(\hat{\mathbb{P}} \in \mathcal{P}', \bar{\mathbb{P}}_i)|M_k] \tag{7-2}$$

where the condition $|M$ denotes that the predicted best source domain $\hat{\mathbb{P}}$ is chosen by the model M. Now the t-test assumes this difference to be distributed normally, $\delta(M_1, M_2) \sim \mathcal{N}(\mu, \sigma^2)$. With this distribution, we can calculate the probability of observing model M_j performing better than or equal to model M_k and vice versa. We choose a p-value of .05 to establish a significant result.

I order to be able to compare two results with the t-test, we first check if the normality assumption $\delta(M_1, M_2) \sim \mathcal{N}(\mu, \sigma^2)$ holds. If not, even if the p-value of significance is lower than 0.05, we cannot say with certainty that the result is a significant and we will therefore not claim that it is.

Single domain selection

First we look at the significance of the results shown in the last column of 7-1. We compare if one model is better in selecting one source domain, i.e. lower error, than another. In Table 7-2 we show the p-values of the normality assumption. For example, we see that the difference between KLD and Chi2 is significantly not normally distributed. In Table 7-3, we showed the p-value determining if the one domain is significantly better performing than the other in terms of cross domain classification error. For example, looking at Table 7-2, we see that the difference between the cross domain classification error for CMEK and KLD selecting the single best domain over 13 runs, is likely to be normally distributed ($p = 0.77 > 0.05$). However, from Table 7-3 we do not see a significant difference between the two ($p = 0.37 > 0.05$).

To evaluate significance of selecting one of the best domains or one of the five worst, we used a binomial distribution. Results are presented in Appendix B, Table B-2 and B-3.

Multiple domain selection

If we let the model select multiple training domains, we are interested in the significance of the cross domain classification error over the range of $n = \{2, 3, \dots, 11\}$. We define the difference vector for the paired t-test

$$\delta(M_j, M_k) = \{\delta_1, \delta_2, \dots, \delta_{11}\} \qquad \delta_n = [\xi(\cup_{z=1}^{n} \hat{\mathbb{P}}_z \in \mathcal{P}', \bar{\mathbb{P}}_i)|M_j] - [\xi(\cup_{z=1}^{n} \hat{\mathbb{P}}_z \in \mathcal{P}', \bar{\mathbb{P}}_i)|M_k]$$

where $\cup_{z=1}^{n} \hat{\mathbb{P}}_z$ denotes the set of n selected training domains according to the selection model. We can look at this evaluation as evaluating if the lines in Figure 7-2 differ significantly. Table 7-4 shows if the normality assumption is met, and Table 7-5 shows the significance of the difference.

In Figure 7-2, each line consists of 10 data points. Each data point is actually an average over 13 runs, each run with a unique target domain. We can evaluate the significant difference of each point between two models. So, in the previous paragraph, we evaluated significance over the entire range $n = \{2, 3, \ldots, 11\}$, we now look if there is a significant difference between models in selecting n domains. We perform a paired t-test over the 13 runs that construct each data point in Figure 7-2. In Appendix B, Tables B-4 to B-9 show the p-values of the normality assumption and significance of the difference.

Table 7-2: P-values of normality assumption of $\delta(M_j, M_k)$ for results selecting 1 source domain.

Selection model	Random	CMEK	Chi2	MMD	EMD	KLD
Random	x	0.81	0.61	0.78	0.64	0.71
CMEK	0.81	x	0.54	0.50	0.62	0.77
Chi2	0.61	0.54	x	0.092	0.065	0.012
MMD	0.78	0.50	0.092	x	0.00085	0.10
EMD	0.64	0.62	0.065	0.00085	x	0.00056
KLD	0.71	0.77	0.012	0.10	0.00056	x

Table 7-3: P-values of significant difference in results of M_1 and M_2 in selecting 1 source domain.

Selection model	Random	CMEK	Chi2	MMD	EMD	KLD
random	x	0.0033	0.078	0.0053	0.000047	0.0019
CMEK	0.0033	x	0.059	0.23	0.16	0.37
Chi2	0.078	0.059	x	0.14	0.0073	0.023
MMD	0.0053	0.23	0.14	x	0.022	0.14
EMD	0.000047	0.16	0.0073	0.022	x	0.22
KLD	0.0019	0.37	0.023	0.14	0.22	x

Table 7-4: P-values of normality assumption of $\delta(M_j, M_k)$ for results selecting multiple source domains.

Selection model	Random	CMEK	Chi2	EMD
Random	x	0.85	0.68	0.31
CMEK	0.85	x	0.0017	0.60
Chi2	0.68	0.0017	x	0.0014
EMD	0.31	0.60	0.0014	x

Table 7-5: P-values of significant difference in results of M_1 and M_2 for selecting multiple source domains.

Selection model	Random	CMEK	Chi2	EMD
Random	x	0.013	0.00043	0.00014
CMEK	0.013	x	0.50	0.25
Chi2	0.00043	0.50	x	0.12
EMD	0.0.00014	0.25	0.12	x

Chapter 8

Conclusion

To select a suitable source domain in terms of cross domain classification error, we hypothesized that this error can be predicted by a function of the source domain distribution $\mathbb{P}$ and marginal target domain distribution $\bar{\mathbb{P}}_x$. We proposed the CMEK model to select the best source domain(s) from a set of candidates. The CMEK model uses a linear combination of the Chi2, MMD, EMD, KLD, the inner domain classification error and the constant 1, with weight vector β as predictor for the cross domain classification error. To evaluate performance, we consider an example including 12 distinct domains forming 132 different source-target domain pairs. The proposed metric parameter β is optimized to minimize the absolute error of the prediction. The optimized predictor was used to select one or multiple best source domains among those 12 domains in order to train a classifier for a thirteenth, unseen target domain. This classifier is tested by calculating the cross domain classification error of the selected source domain(s) and the target domain. The process is repeated 13 times, each time using a different unseen target domain to test on. We benchmark performance with randomly selecting a source domain and with using all candidate source domains to train on.

From figure 7-1 we see that the CMEK model is well able to identify source domains with low cross domain classification error. In 54% of the runs, the selected source domain was within 5 percent points error of the optimal choice whereas random selection only selected 16% of the times a source domain in this category.

From Table 7-1, we see that compared to random domain selection, the CMEK, MMD, EMD and KLD selection realize significant improvement in average cross domain classification error ($p = 0.0034, p = 0.0054, p = 0.000047$ and $p = 0.0019$ respectively). The CMEK model has a significant larger probability of selecting the best domain compared to random, MMD and Chi2 selection ($p = 0.0029, p = 0.0021$ and $p = 0.038$ respectively). If we look at the probability of not selecting one of the five worst domains for training, all models except the Chi2 selection model significantly perform better than the random selection model. From (7-1) we see that the CMEK model depends primarily on the MMD and the inner domain classification error.

Although most selection models significantly improve performance compared to random selecting one source domain, we would still be better off training on all the available source domains. However, if we let our CMEK model select multiple source domains to train on, we are able to get better performance than training on all source domains for some n, see Figure 7-2. The CMEK model seems to perform significantly better than training on all domains when it selects 8 or 9 domains ($p = 0.047$ and $p = 0.041$). Note, for all mentioned significant results, the normality assumption of the t-test is confirmed, however in this specific evaluation, the pairwise differences seem not normally distributed. What the optimal number of domains to train on is, may be very dependent on what candidate source domains are available. When we have candidate source domains that are somehow similar to the target domain, the optimal n will be higher. For a diverse set of candidate source domains the optimal n will be lower.

Looking closely at the curve of the random domain selection for n training domains, we can conclude that we are still not at a saturation point; adding more data to train on will still improve performance. Making a logarithmic projection for adding more domains to train on, we expect the the cross domain classification error to be .27 for 20 training domains and .22 for 50 training domains. However, confidence intervals are quite large on these predictions. See see Appendix B for the projection.

Also, from Figure 7-2, we see that the EMD selection really loses its magic when multiple domains have to be selected. Apparently, the EMD distance is primarily good in detecting the few best domains, and much less capable of estimating the cross domain classification error of medium range source domains. The CMEK model performs significantly better than EMD and random selection in selecting multiple source domains to train on ($p = 0.000023$ and $p = 0.046$ respectively). The CMEK model is for each $3 \geq n \leq 11$ significantly better then random selection. Compared to the CMEK model, EMD selection significantly fails in selecting 6, 7 and 8 source domains.

If we are to make a general recommendation for using one of the evaluated models, in the light of general performance, we would choose the CMEK model. The CMEK model shows significantly good performance and stable behavior in selecting multiple source domains and it has solid performance in selecting the single best domain. In retrospect to our hypothesis (5-4), we can conclude that it is to some extend possible to approximate the cross domain classification error as we were able to use this approximation for successful source domain selection. However, there is much more room for improvement.

Chapter 9

Further discussion

In this chapter, we reflect on the proposed CMEK model and our recommendation. We will elaborate on the weaknesses of our approach and will give some context information in order to be able to assess implementation of the CMEK model on other data sets.

9-1 Discussion

Reflecting on our conclusion, we would like to place some remarks. We showed that the CMEK model works superior in selecting one source domain compared to random selection. However, the results show that, even for candidate source domains with a wide spread in topic and source medium, it is quite beneficial to train on all the candidate source domains. One of the reasons not to train on all data could be that it is too computational expensive. Another reason might be that the candidate source domains are too diverse. To establish if this is the case, we would need a measure that informs us about the diversity of the candidate source domains.

When we know we have candidate source domains that are similar to each other in terms of expressing sentiment, it might very well be that our CMEK model is not able to improve performance compared to training on all data. Therefore, if we have candidate source domains that are similar to each other, we might choose to train on all domains for simplicity.

Another disadvantage of the CMEK model, is that it uses some distance measures that are quite expensive to calculate. This can be a problem when we have many or large candidate source domains. This challenge might be addressed by using less features and more informative features to calculate distance over and we could optimize used code on efficiency. We might investigate what happens to the performance of the CMEK model in case we leave out the Chi2 and KLD measure, as they are given fewest importance, i.e. low weight β_i.

We constructed, tested and evaluated the model with the 13 acquired data sets. We would like to remind that sentiment expression is characterized by a severe notion of stochasticity. It is hard to tell how much performance of the CMEK model will deviate when 13 other sets

are chosen to construct the model. Furthermore, we have not evaluated the model when it is constructed on only 3 or 4 data sets for example. For more available domains to construct the model, we might assume that the model will be more refined and better performing due to the wider support for the β weight vector.

Although we described a method that can be easily implemented in other fields of machine learning that encounter a domain shift such as computer vision, fraud detection or spam detection, we only showed our model works reasonably well in the domain of SC.

9-1-1 Future work

As the selection models presented in Table 7-1 are to some extension able to select a source domain with lower than average cross domain classification error, we concluded that the measures are to some extend able to predict the cross domain classification error in order to select a suitable source domain. However, if our model would fully do justice to our hypothesis (5-4), we would improve even further. We could say that proof of concept is given: it is possible to roughly predict the cross domain classification error based on the distribution of the source domain distribution and marginal distribution of the target domain. However, we still see much room for improvement as the lower bound for the error when selecting one domain is .252, we're only half way.

If we would like to improve results further, first steps would be to look closer into the features over which we measure distance. Should we use the 1000 most common, or is distance better measured with more or less features? This could be dependent on corpus size as well. There might be feature selection methods to select the features that are most informative in terms of distance, instead of simply using the N most occurring features.

To improve, we might add some predictors to the linear combination, such as the document length distribution or the length of a corpus. Brief statement are on average more similar with each other than short statements compared with long expressions. Also, if we have more objects in a source domain, we might benefit more when training on that large source domain than training on a corpus with only a few documents. However, with more elements in β, we might need more data to prevent over fitting.

We can also extend our model to give a weight factor on every source domain: source domains that are predicted not suitable get a low weight, domains with low distance to the target are given more weight. This approach leans towards importance sampling.

We constructed a predictor for the cross domain classification error of one source domain, and used the model to select multiple source domains. If this is what we are interested in, it would make sense to calculate distances between a set of multiple domains and the target domain and make a selection model optimized on predicting the cross domain classification error when training on multiple domains. In that case, synergies of data sets will be taken into account. This approach will, however, be very computational expensive.

Appendix A

Final paper

Distance Based Source Domain Selection for Automated Sentiment Classification

Lex Razoux Schultz
Delft Center for Systems and Control
Delft University of Technology
2628CC Delft, the Netherlands

Marco Loog
Pattern Recognition Laboratory
Delft University of Technology
2628CD Delft, the Netherlands

Peyman Mohajerin Esfahani
Delft Center for Systems and Control
Delft University of Technology
2628CC Delft, the Netherlands

***Abstract*—Automated sentiment classification (SC) on short text fragments has been an upcoming field of research. Performing SC in unseen domains with few or no labeled samples can significantly affect the classification performance and is likely to results in an adaptation loss due to different expression of sentiment in source and target domain. We evaluate the effectiveness of using an unsupervised measure to identify a suitable source domain for an SC task in a specific target domain. A linear combinations of the Chi squared distance, Maximum Mean Discrepancy, Earth Mover's Distance and Kullback-Leibler Divergence to select a source domain will be assessed. Results show that selecting a source domain by using this linear combination and the inner domain classification error of the candidate source domain, results in a reduction of adaptation loss by 7 percent points compared to training on a randomly selected source domain. The model is proposed as the CMEK selection model. In addition, the CMEK model is able to select a subset of all candidate domains to train on and realize slightly improved performance compared to training on all candidate source domains. Therefore, for an SC task of a given target domain, when no labeled target domain data is available, we propose to use the novel CMEK selection model for choosing one or multiple source domains to train on.**

***Index Terms*—Sentiment classification, sentiment analysis, domain shift, distance measure, source selection**

I. Introduction

Automated SC is performed when a trained classifier is used to label documents with a sentiment label, based on the content of the document. In practice, a document represents a review, tweet or other small textual opinion expression and oughts to be classified as either positive or negative (binary SC). Allied terminology for SC is semantic orientation or polarity, which indicates the direction a word deviates from the norm of its semantic group [1]. Often, the terms opinion mining and sentiment analysis are used to describe the computational treatment of opinion, sentiment and subjectivity in text [2]. Examples of current applications that benefit from SC are recommendation systems [3], stock market prediction systems [4] and political election predictors [5]. Also in the future, we could imagine SC put in practice to improve man-machine communication such as voice commands or even interaction between robots and humans.

One of the biggest challenges in many SC applications, is the discrepancy between the source domain where the classifier is trained on, and the target domain of interest. Typically, we have large amounts of labeled data from different source domains, but only possess few, unlabeled data from the target domain, both originated from different underlying distributions. In this setting, SC generally performs rather bad compared to inner domain training and testing. This so called domain transfer problem can be intuitively explained due to different fashions of sentiment expression in different domains [6]. Various research has been performed with the aim to decrease the loss in classification performance due to domain shift. This loss is referred to as the adaptation loss [7] [8]. One of the popular transfer learning techniques to maintain performance of SC when crossing domains, is Structural Correspondence Learning which uses pivot features as link between the source and target domain [9] [10]. Many other techniques are based on optimal transport which matches the conditional probability distribution of the training domain and target domain by a transformation of the source domain data. The distance between source and target distribution is minimized [11]. Often distance is minimized in a lower dimensional embedded space, for example when performing Subspace Alignment [12] or Transfer Component Analysis [13]. Other approaches successfully use unsupervised feature scaling to increase performance when source and target domain differ in topic or author group [14].

Another approach to reduce adaptation loss that has received less attention in literature, is unsupervised source domain selection. This selection method attempts to select an optimal source domain, for training purpose, from a set of candidate domains without supervision; i.e. it selects the source domain that has best cross domain classification performance compared to the other candidates without the use of class labels of the target domain data. Source domain selection has been successfully used before, for example to improve brain-computer interface calibration. In this case, supervised source domain selection based on distance between the class average vectors has been used [15].

In the field of source domain selection for SC, Blitzer et al. show a good correlation between the $\mathcal{A}$-distance and adaptation loss [8]. However, the adaptation loss is measured using the target domain's inner domain classification error which requires labeled data of the target domain. The concept of $\mathcal{A}$-distance is also used in recent research for orthophoto

classification, were it is approximated by the Maximum Mean Discrepancy [16]. For even more open search space for source domain, techniques have been researched to find a suitable source domain in open online information sources such as Wikipedia [17]. However, this techniques also uses labeled information of the target domain. In this paper, we propose a new method, CMEK, for source domain selection that does not require any labeled information of the target domain.

In the second section of this paper, we define the problem of source domain selection in general, that is, not in context of SC. Next, in the method section, we propose an approach to predict the performance of a classifier trained on a source domain and applied on a target domain of interest of which the labels are unknown in order to find a solution for the defined problem. This prediction is based on statistical distances between the distributions of the source and target data. We use several statistical measures to determine the distance, which will be elaborated on. We will address certain limitations to our proposed model, starting with fundamental limitations that occur in all classification tasks and ending with limitations of our approach within the task of SC. Section V describes the experimental setup in detail and will elaborate on the data sources we used and the model settings we chose. In section VII we will objectively present results. In the conclusion we reflect on the results and offer suggestions for future work.

II. Problem definition

With a document represented by a random variable X and its sentiment label by random variable Y, each domain is uniquely characterized by its joint probability density function supported on $\mathbb{X} \times \mathbb{Y}$. Let us then define two underlying distributions to the evaluated source and target data on $\mathbb{X} \times \mathbb{Y}$, denoted by $\mathbb{P}$ and $\bar{\mathbb{P}}$ respectively. Their marginals on $\mathbb{X}$, are denoted as $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$ respectively. In machine learning, a realization of $\mathbb{P}$ is referred to as labeled data whereas realizations from $\mathbb{P}_x$ are referred to as unlabeled data.

We define an hypothesis function as a mapping from $\mathbb{X}$ to $\mathbb{Y}$, $h : \mathbb{X} \to \mathbb{Y}$. In the context of classification problems in the machine learning literature, the set $\mathbb{Y}$ is often discrete, often binary, and the hypothesis h is often referred to as a classifier as it is able to assign a class label in $\mathbb{Y}$ to each data point in $\mathbb{X}$. The true hypothesis function $h^\star_{\mathbb{P}}$ for a domain characterized by $\mathbb{P}$, is defined as

$$h^\star_{\mathbb{P}} := \arg\min_h P\big(h(x) \neq y \mid (x, y) \sim \mathbb{P}\big). \tag{1}$$

We then define the inner domain classification error as the probability of misclassification in the same domain as the true hypothesis function is constructed in. And the cross domain classification error is defined as the probability of misclassification in the target domain of the true hypothesis function of the source domain

$$\begin{aligned} \xi(\mathbb{P}) &:= P\big(h^\star_{\mathbb{P}}(x) \neq y \mid (x, y) \sim \mathbb{P}\big). \\ \xi(\mathbb{P}, \bar{\mathbb{P}}) &:= P\big(h^\star_{\mathbb{P}}(x) \neq y \mid (x, y) \sim \bar{\mathbb{P}}\big). \end{aligned} \tag{2}$$

Note that due to the stochastic nature of sentiment expression, $\xi(\mathbb{P})$ and $\xi(\mathbb{P}, \bar{\mathbb{P}})$ are not likely to be zero, even if we would have infinite data to train on. In addition, due to the difference in source and target distribution, $\xi(\mathbb{P}, \bar{\mathbb{P}})$ is likely to be higher than $\xi(\mathbb{P})$ [8]. The difference between the cross and inner domain classification error is referred to as the adaptation loss.

The goal in many classification tasks is to minimize the cross domain classification error (2) for a specific target domain while we have access to a finite set of labeled candidate source domains. An appropriate choice of the source domain in this context is the main goal of this study, leading to the following question:

What source domain minimizes the cross domain classification error for a given target domain?

With the choice of source domain restricted to a domain characterized by a probability density function in the candidate set $\mathcal{P}$, our main goal can be formally described through the optimization program

$$\mathbb{P}^\star := \arg\min_{\mathbb{P} \in \mathcal{P}} \xi(\mathbb{P}, \bar{\mathbb{P}}). \tag{3}$$

where $\xi(\mathbb{P}, \bar{\mathbb{P}})$ is the cross domain classification error introduced in (2). In other words, when we posses a finite number of labelled data sets, each originated from a unique source domain, on which set should we train our classifier in order to minimize classification error on data from a specific target domain?

The challenge concerning the objective (3), is that for a target domain of interest we typically only have unlabeled data, therefore we cannot calculate the cross domain classification error. This means we only know the marginal distribution $\bar{\mathbb{P}}_x$ of our target domain instead of $\bar{\mathbb{P}}$, making it impossible to explicitly calculate $\xi(\mathbb{P}, \bar{\mathbb{P}})$. In order to find the best source domain characterized by $\mathbb{P}^\star \in \mathcal{P}$ for our target domain with distribution $\bar{\mathbb{P}}$, we ought to predict $\xi(\mathbb{P}, \bar{\mathbb{P}})$ for all $\mathbb{P} \in \mathcal{P}$. For this prediction we have full distributional information of the source domains, $\mathbb{P}$, and marginal information on the distribution of the target domain, $\bar{\mathbb{P}}_x$, available.

III. Method

In this section, we propose a model, called CMEK source domain selection, an acronym of the four measures it uses, to deal with the previously stated challenge. The model measures statistical distances between the marginal distributions of candidate source domains and the target domain and uses the inner domain classification error of the candidate source domains. The candidate source domain with the lowest distance is

hypothesized to have the lowest cross domain classification error and is selected to train a classifier. This section will describe the CMEK model, the distance measures it uses and how we will evaluate the performance of the model.

We consider a set $\mathcal{D}$ containing a family of distance functions $d : \mathbb{P} \times \bar{\mathbb{P}}_x \to \mathbb{R}_+$. We now hypothesize that given an available set of source domain distributions $\mathcal{P}$ and a target domain distribution denoted by $\bar{\mathbb{P}}$ with the marginal $\bar{\mathbb{P}}_x$, there exists a $\hat{d} \in \mathcal{D}$ that can reliably predict the cross domain classification accuracy $\xi(\mathbb{P}, \bar{\mathbb{P}})$, that is,

$$\xi(\mathbb{P}, \bar{\mathbb{P}}) \overset{\text{hyp}}{\approx} \hat{d}(\mathbb{P}, \bar{\mathbb{P}}_x). \tag{4}$$

More specific, we hypothesize that the cross domain classification error can be predicted by looking at the difference in marginal distribution functions $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$ and $\xi(\mathbb{P})$. With this prediction, the best candidate source domain can be selected for training. We formalize this optimal choice of the measure by considering the optimization problem

$$\hat{d} := \arg\min_{d \in \mathcal{D}} \sum_{\mathbb{P} \in \bar{\mathcal{P}}} \left| \xi(\mathbb{P}, \bar{\mathbb{P}}) - d(\mathbb{P}, \mathbb{P}_x) \right|. \tag{5}$$

To construct the family $\mathcal{D}$ of candidate measures, we use a vector s with K known statistical distance measures as elements, $s_i : \mathbb{P}_x \times \bar{\mathbb{P}}_x \to \mathbb{R}_+$ for $i \in \{1, \cdots, K\}$. The family $\mathcal{D}$ consists of linear combinations of these measures s_i with corresponding weight coefficients $\beta_i \in \mathbb{R}_+$, with β_i being elements of the weight vector β. We augment the linear combination with a constant. Note that the distance measures are functions supported on the marginal distributions of source and target domain. Examples of these statistical distance metrics are the Chi squared distance (Chi2), Maximum Mean Discrepancy (MMD), Earth Mover's Distance (EMD) and Kullback-Leibler Divergence (KLD). When a source domain has poor inner domain classification performance, we expect it to be less suitable as source domain for cross domain classification. Therefore, we calculate the inner domain classification error for all candidate source domains and add this to our vector s. With our choice for $\mathcal{D}$, the optimization problem (5) can be reduced to

$$\begin{aligned} \hat{\beta} &:= \arg\min_{\beta \geq 0} \sum_{\mathbb{P} \in \mathcal{P}} \left| \xi(\mathbb{P}, \bar{\mathbb{P}}) - \beta s(\mathbb{P}, \bar{\mathbb{P}}_x) \right|, \\ \hat{d} &:= \hat{\beta} s(\mathbb{P}, \bar{\mathbb{P}}_x). \end{aligned} \tag{6}$$

Note that program (6) constructs the optimal unsupervised predictor by using the full distribution of the target data, which is presumed to be unknown. In a practical setting with a finite number of elements in $\mathcal{P}$, we deal with this problem by training only on the domains in $\mathcal{P}$. In each run, one of the elements of $\mathcal{P}$ is extracted from the set and used as proxy $\bar{\mathbb{P}}$, at the end of the run, this element, denoted as $\tilde{\mathbb{P}}$ and its marginal $\tilde{\mathbb{P}}_x$, is placed back in $\mathcal{P}$. In every run, another element is extracted and placed back for training. This training program is described as

$$\hat{\beta} := \arg\min_{\beta \geq 0} \sum_{\mathbb{P} \in \mathcal{P}, \mathbb{P} \neq \tilde{\mathbb{P}}} \sum_{\tilde{\mathbb{P}} \in \mathcal{P}} \left| \xi(\mathbb{P}, \tilde{\mathbb{P}}) - \beta s(\mathbb{P}, \tilde{\mathbb{P}}_x) \right|.$$

We hypothesize that with a sufficient amount of source domain distributions in $\mathcal{P}$, the constructed predictor for the cross domain classification error, based on information from $\mathcal{P}$, is also reasonably accurate in predicting the cross domain classification error for a target domain not included in $\mathcal{P}$. This allows us to select the best source domain to train on for this unseen target domain.

Now, in view of our hypothesis (4), with the predictor $\hat{d}$ we are able to approximate $\xi(\mathbb{P}, \bar{\mathbb{P}})$ with only marginal information of the target domain $\bar{\mathbb{P}}_x$, which allows us to deal with the challenge through an approximation. This enables us to, given a target domain with distribution $\bar{\mathbb{P}}$, select the source domain, characterized by $\hat{\mathbb{P}}$ from a set of available source domains that is predicted to minimize the cross domain classification error $\xi(\mathbb{P}, \bar{\mathbb{P}})$ as formalized through the optimization program

$$\hat{\mathbb{P}} := \arg\min_{\mathbb{P} \in \mathcal{P}} \hat{d}(\mathbb{P}, \bar{\mathbb{P}}_x)$$

which is the goal of this study.

A. Measures

For our SC case study, we hypothesize that the way people express sentiment can be captured in how often certain words and word combinations are used. The underlying joint distributions $\mathbb{P}$ differs among different domains. To accurately classify sentiment for a target domain, we therefore ought to find a domain with similar underlying distribution function. But how do we measure similarity between distributions?

When we talk about statistical similarity or distance measures, we can define two different approaches of measuring distance. First, we can compare statistical meta data such as the average word frequency. The Chi2 distance is based on comparing this average word frequency between two evaluated corpora. Other meta approaches are, for example, based on comparing rank of word frequency. These approaches are in general computational cheap, but not very robust. Another approach is to compare the distribution of word frequency. These approaches are referred to as Integral Probability Metrics (IPMs). Where the previously described metrics only compare meta data on the word frequency distributions, IPMs compare the actual distributions. IPMs use a class $\mathcal{F}$ of functions $f : \mathbb{X} \to \mathbb{R}$ to calculate the distance s_i between $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$ as

$$s_i(\mathbb{P}_x, \bar{\mathbb{P}}_x) = \sup_{f \in \mathcal{F}} \left| \mathbb{E}^{\mathbb{P}_x}\big[f(X)\big] - \mathbb{E}^{\bar{\mathbb{P}}_x}\big[f(X)\big] \right|, \tag{7}$$

the question arises which class $\mathcal{F}$ to choose [18]. If we would choose the 1-Lipschitz function for $\mathcal{F}$, we get the popular and straightforward EMD metric [19]. This metric is straightforward as it measures how much distribution mass has

to be moved to transform one distribution in another. The mass is weighted proportionally to the distance it has to travel. If we would choose $\mathcal{F}$ in (7) to be a class of functions in the unit ball of a reproducing kernel Hilbert space, we would have the MMD metric [20]. This approach finds a well behaved, smooth, function which is high on the points drawn from $\mathbb{P}_x$ and low on $\bar{\mathbb{P}}_x$ or vice verse. We use as our test statistic the difference between the mean function values on the two samples according to (7) [21]. Another approach is to choose a logarithmic function for f, giving the KLD with the interesting asymmetry property $s_i(\mathbb{P}_x, \bar{\mathbb{P}}_x) \neq s_i(\bar{\mathbb{P}}_x, \mathbb{P}_x)$. As difference in distribution becomes larger, KLD gives decreasingly increasing distance.

Since we hypothesize that the way people express sentiment can be captured in how often certain words and word combinations are used, but not know in what relationship, we propose a broad spectrum of possibilities to optimize over. We propose to evaluate a meta approach, Chi2 distance, and three IPM approaches that use different comparison functions f: MMD, EMD and KLD. Therefore we have four measures, a constant and the inner domain classification error of the source domain that span the distance vector space $\mathcal{D}$. In the next paragraphs we will briefly elaborate on how we use these distance measures in a discretized fashion. Selection based on the linear combination of these four metrics, the constant and $\xi(\mathbb{P})$ will be referred to as the CMEK selection model.

1) Chi2: As first distance measure, we make use of the test statistic of Pearson's χ^2 test [22]. Let us have two unlabeled corpora, c_x and $\bar{c}_x$, containing the documents of the source and target domain respectively. Then, let $p(w)$ be the probability of a word in c_x being w, and $\bar{p}(w)$ indicating the probability of a word being w in $\bar{c}_x$. We calculate the value of $Chi2(c_x, \bar{c}_x)$ as the sum of Pearson's test metric over the N most occurring words, $w_1, w_2, \ldots, w_N$,

$$Chi2(c_x, \bar{c}_x) := \sum_{i=1}^{N} \frac{\big((p(w_i) - \bar{p}(w_i)\big)^2}{p(w_i) + \bar{p}(w_i)}.$$

2) MMD: Our second distance measure is the Maximum Mean Discrepancy (MMD) defined as the largest difference in expectations over functions in the unit ball of a reproducing kernel Hilbert space [20]. Let a unit ball $\mathcal{F}$ in a universal reproducing kernel Hilbert space $\mathcal{H}$ be a class of functions $f : \mathbb{X} \to \mathbb{R}$ and $\mathbb{E}^{\mathbb{P}}[X]$ be the expected value of random variable X distributed according distribution $\mathbb{P}$. Then the MMD is defined as

$$MMD(\mathbb{P}_x, \bar{\mathbb{P}}_x) := \sup_{f:||f||_{\mathcal{H}} \leq 1} \Big| \mathbb{E}^{\mathbb{P}_x}\big[f(X)\big] - \mathbb{E}^{\bar{\mathbb{P}}_x}\big[f(X)\big] \Big|. \tag{8}$$

3) EMD: The Earth Mover's Distance (EMD), similar to the Wasserstein metric [23], is a broadly used metric to compare distributions. The metric measures how much probability mass has to be moved, and how far it has to be moved, to transform two distributions into each other. Imagine two discrete distribution functions $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$ with cluster representatives b and $\bar{b}$ both with m clusters and their corresponding probabilities p and $\bar{p}$ adding both up to one. This can be visualized by a histogram with m bins, centered on b, with heights p, and a similar representation for the target domain data. Then, we define a distance matrix $D = [d_{i,j}] \in \mathbb{R}^{m \times m}$, indicating the distances between the two cluster representatives b and $\bar{b}$. At last we define the flow matrix $F = [f_{i,j}] \in \mathbb{R}^{m \times m}$ as the flow of probability mass that has to be moved to transform the two distribution into each other. The EMD value between the two distributions is the minimized weighted, by D, flow needed for transformation,

$$\begin{aligned} EMD(\mathbb{P}_x, \bar{\mathbb{P}}_x) := \quad & \min_{F} \sum_{i=1}^{m} \sum_{j=1}^{m} d_{i,j} f_{i,j} \\ \text{subject to} \quad & f_{i,j} \geq 0 \\ & \sum_{i=1}^{m} f_{i,j} = p_i \\ & \sum_{j=1}^{m} f_{i,j} = \bar{p}_j. \end{aligned}$$

4) KLD: As last measure, we use the Kullback-Leibler Divergence (KLD) [24], also known as relative entropy [25]. For two discrete probability functions $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$, the natural logarithm of the ratio of their two probability values p and $\bar{p}$ respectively are basis for the KLD value. With cluster representatives $b = \bar{b}$ and m clusters, the KLD value is defined as

$$KLD(\mathbb{P}_x, \bar{\mathbb{P}}_x) := \sum_{i=1}^{m} p_i \log\left(\frac{p_i}{\bar{p}_i}\right).$$

Notice that this metric is asymmetric which might come in handy as the cross domain classification error is asymmetric as well.

B. Performance evaluation

We have constructed a model that predicts cross domain classification error between a candidate source domain and a target domain with a linear combination of statistical distance measures based on the marginal distributions $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$, a constant and $\xi(\mathbb{P})$. Now we are interested in how well this model is able to identify the best candidate source domain, i.e. the one that gives the lowest cross domain classification error $\xi(\mathbb{P}, \bar{\mathbb{P}})$. For n_c acquired data sets, let us use one corpus as target domain, $\bar{\mathbb{P}}$, and the others as candidate source domains with the set annotation $\mathcal{P}'$. We can repeat the experiment n_c times by using each time a different target domain and average the results.

We first let the model select the predicted best source domain $\hat{\mathbb{P}} \in \mathcal{P}'$ and use this domain for training. To evaluate performance we define the relative cross domain classification error as

$$\xi_{relative}(\hat{\mathbb{P}}, \bar{\mathbb{P}}) = \xi(\hat{\mathbb{P}}, \bar{\mathbb{P}}) - \xi(\mathbb{P}^\star, \bar{\mathbb{P}}) \tag{9}$$

where $\mathbb{P}^\star$ is the domain in $\mathcal{P}'$ of which we know it has the lowest cross domain classification error, the true best domain. When the model selects the best source domain, the relative error will therefore be zero. We perform the same analysis when using each measure individually as predictor. We calculate the average relative error but also look at the distribution of this relative error to see how often our models selects one of the best candidate source domains and how often it fails and selects a poor candidate source domain.

Next, we let our proposed CMEK model select the best n domains from $\mathcal{P}'$ and compare $\xi(\bigcup_{i=1}^{n} \hat{\mathbb{P}}_i, \bar{\mathbb{P}})$ with training on all domains in $\mathcal{P}'$. We perform the same analysis when using the Chi2 distance and EMD measure individually. All results will be tested on significance using a paired t-test over the results of 13 runs, we will use a significance level of $p = 0.05$. The paired t-test is strongly dependent on the assumption that the pairwise differences are normally distributed. For every comparison between results, we evaluated the normality assumption. We report only on significant results for which the normality assumption is confirmed.

IV. Limitations

In solving the problem as defined in section II with our proposed method, we encounter limitations of various kinds. We first discuss a fundamental statistical limitation that is present in almost all practical machine learning problems. Then we discuss a limitation of computational nature and how to deal with this. In the last paragraph we elaborate on limitations that are more specific to the case study of SC: numerically representing text and the limited hypothesis space. We end with briefly explaining the limitations of our proposed CMEK model.

A. Statistical limitations

In the first place, and inherent to all classification problems, we are unable to retrieve the true distribution over the support set $\mathbb{X} \times \mathbb{Y}$ in practical settings. Instead, we need to infer this distribution through only a finite number of observations. In this case a natural approximation for the distribution is the empirical distribution supported on these observations, which are the available documents in our SC case study. We then encounter an inevitable approximation

$$\mathbb{P} \approx \frac{1}{n_s} \sum_{i}^{n_s} \delta_{\{x_i, y_i\}}$$

where n_s denotes the number of documents in the domain sample. Due to the approximation of the true distribution, the hypothesis function (1) will be suboptimal, resulting in a higher classification error. To evaluate this error, the available data originated from one domain is split into a train and test set. The train set is used to construct $h^\star_{\mathbb{P}}$ as defined in (1). The test set is used to calculate the inner domain classification error. When using sufficient data in the test set, a large difference in apparent and true error indicates that the observations used for training are a poor support for approximating the true distribution since it means that the empirical distributions from the train and test set are not much alike. When this is the case, we are over fitting; the hypothesis function works for the data it is constructed on, but can not be properly generalized to perform well on new data from the same underlying distribution. To reduce over fitting, we ought to use a sufficient number of training objects and a low but not too low number of features.

B. Computational limitations

Another limitation is that the optimization in (1) uses the indicator function as loss function, which is non-convex. For computational reasons, we replace this loss function with a convex counterpart. For the true hypothesis function $h^\star_{\mathbb{P}}$ in (1), a common convex loss function $\ell(x, y)$ to minimize, is the quadratic error of the prediction, popular as the method of least squares

$$h^\star_{\mathbb{P}} = \arg\min_h \sum_{i=1}^{n_s} \ell(x_i, y_i) \qquad \ell(x, y) := \big(h(x) - y\big)^2, \quad (10)$$

and the cross domain classification error $\xi(\mathbb{P}, \bar{\mathbb{P}})$ will be defined by empirical distributions of the target and the hypothesis function from (10),

$$\xi(\mathbb{P}, \bar{\mathbb{P}}) := \frac{1}{n_s} \sum_{i}^{n_s} \mathbb{1}_{\left\{h^\star_{\mathbb{P}}(x_i) \neq y_i \quad | \quad (x,y) \sim \bar{\mathbb{P}}\right\}}. \qquad (11)$$

C. Further discussion on SC related limitations

There are limitations concerned with performing SC with a nature that may also be relevant to other fields of application. Choosing a word representation model is a field of study by itself and should be considered carefully here. In the previous section we assumed a true hypothesis function $h^\star$ mapping each document represented in $\mathbb{X}$ to a label in $\mathbb{Y}$. For computational purpose, we need to numerically express documents in the $\mathbb{X}$ space. Let us call W the set of all words, $W = \{w_1, w_2, \dots w_{n_w}\}$ where n_w is the number of unique words. Then, documents in $\mathbb{X}$ are combinations of these words, indicating that $\mathbb{X}$ can be viewed as an element in the power set of W denoted by $\wp(W)$. Note that repetition in the subset is possible. For computational feasibility, we choose a subset $\mathcal{F} \subset \wp(W)$ to represent our documents. The subset $\mathcal{F}$ is called the feature set whose cardinality is denoted by $|\mathcal{F}|$. The features selection process depends on what representation model is at hand. The representation model $\mathfrak{F}$ maps $\mathbb{X}$ to $\mathbb{R}^{|\mathcal{F}|}$, i.e. $\mathfrak{F} : \mathbb{X} \to \mathbb{R}^{|\mathcal{F}|}$. The computational need for a representation model $\mathfrak{F}$ to select a subset of $\mathbb{X}$ for representation, limits the classification model since information has to be discarded, see Figure 1.

Furthermore, in the task of classification we need to define a search space $\mathcal{H}$ for our hypothesis function as it is computational impossible to search through all possible hypothesis functions as proposed in (1). SC tasks are characterized by the fact that the representation space is sparse, i.e. the ratio

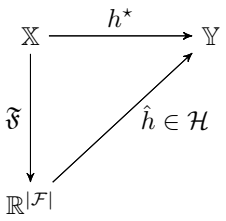


Fig. 1. Representation and classification structure

of number of training documents to $|\mathcal{F}|$ is significantly low. In this situation, computational feasibility calls for a tractable subset of hypothesis functions $\mathcal{H}$ such as all linear functions. The true hypothesis function is not likely to be in our hypothesis space, $h^\star \notin \mathcal{H}$. We define the optimal hypothesis function constructed in the domain $\mathbb{P}$ as the hypothesis $\hat{h}$ within our limited hypothesis space $\mathcal{H}$, which is the collection of all linear mappings from $\mathbb{R}^{|\mathcal{F}|}$ to $\mathbb{R}$, that minimizes the chosen loss function (10). The cross domain classification error from (11) is calculated with the optimal hypothesis function $\hat{h}_{\mathbb{P}}$.

Hereafter, we use $\mathbb{P}$ to annotate the empirical distribution instead of the true continuous distribution. Furthermore, the cross domain classification error defined as $\xi(\mathbb{P}, \bar{\mathbb{P}})$ in (2) is calculated with the optimal hypothesis function in $\mathcal{H}$ that minimizes the empirical loss in (10).

D. CMEK selection model limitations

Our model assumes that a good source domain, with low cross domain classification error, can be identified by the distributions $\mathbb{P}$ and $\bar{\mathbb{P}}_x$. However, where in reality sentiment is determined by joint distribution of the features, the measure we use only measure distance on the individual distributions of the features. This means we try to find similarity of the joint distributions from marginal distributions of individual features. The same holds for classification that is based on marginal data of individual features whereas sentiment in real life is determined by combinations of features. Since we use marginal data for calculating the cross domain classification error as well as the prediction of this error, this approach seems justified, but it creates a model separated from the true fashion of expressing sentiment.

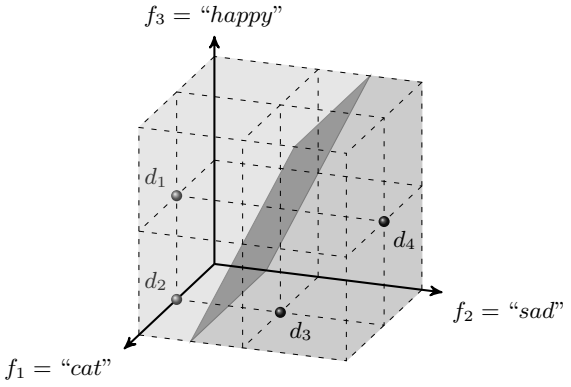
In addition, we do not know in what way we should compare distributions, only that when distributions are the same, we expect no adaptation loss. It might very well be that the linear combination of our chosen measures does not include the true distance function that distinguishes domains from each other.

V. Case study: Sentiment classification

Let us give some brief background information on how to perform SC, mainly to clarify design choices made in the experimental set up. The pipeline for SC can be segmented into three parts: data preprocessing, document representation and classification. The next paragraphs will describe the pipeline. For a lower dimensional visualization, see Figure 2.

A. Preprocessing

In the preprocessing part one tries to remove noise from the text that does not hold sentiment information in order to reduce complexity, i.e. dimensionality. Common techniques to do so are stop word removal, lower casing words, spelling correction and removing punctuation. Other techniques, including part of speech tagging, stemming and lemmatization, attempt to find implicit sentiment information by predicting syntactics of words or combinations of words.



$d_1 =$ "The **cat** makes me **happy**."	POS
$d_2 =$ "The **cat** is beautiful."	POS
$d_3 =$ "The **cat** makes me **sad**."	NEG
$d_4 =$ "I was **happy**, the **sad cat** made me **sad**."	NEG

Fig. 2. Classified feature space

B. Document representation

The second part of the pipeline for sentiment classification deals with representing documents in a mathematical interpretable fashion. The most upfront approach is to use words as features, and word counts as feature values without looking at word order, a bag-of-words approach. With n_F words or combinations of words as features, $n_F = |\mathcal{F}|$, we can build an n_F-dimensional feature space in which each word is represented by an n_F-dimensional vector. The number of occurrences of a feature in the document determines the value in that dimension. Each document is then represented as the sum of all the vectors of its features. When using n_F features, a corpus of n_d documents can be represented as a matrix $X \in \mathbb{N}^{n_d \times n_F}$ which is referred to as the feature value matrix. X represents a projection of the empirical distribution $\mathbb{P}$. In addition one could choose to use combinations of N words as features, called N-grams. Often, features are weighted to assign less value to more common words such as stop words. A widely used weighting scheme borrowed from information retrieval systems is the TF-IDF weighting scheme. To reduce dimensionality, feature selection algorithms can be used, selecting the most class label informative features by using for example χ^2, mutual information or lexicon based selection [26]. Another popular approach for document representation

uses word embeddings [27] [28], a lower dimensional projection of a high dimensional feature space. To construct a good projection, we need a lot of data, preferably from the domain that is under evaluation.

C. Classification

When we have a mathematical representation for the source and target documents, we try to find a hypothesis function that separates the feature space in subspaces belonging to the different class labels. All documents that are represented in certain subspace, are predicted to have the accompanying label of that specific subspace. The hypothesis function, or classifier, is constructed by minimizing a loss function, see (10). Popular classifiers for sentiment classification are Support Vector Machines, Naive Bayes and Logistic Regression (LR).

VI. Experimental setup

In this section we will briefly elaborate on the data sets we used for our experiments and the design choices we made concerning the SC pipeline and used distance measures.

A. Data sets

We acquired 13 data sets, corpora, each consisting of a different number of documents ranging from 502 to 30602 documents with an average of 6326 documents. The average length of each document among the corpora varies between 10 words and 20 words with an average of 15 words. We used a corpus of online reviews on movies, venues and consumer products [29], short opinions about movies [30] and nine sets of tweets with various topics [31]. The acquired data sets do not have any major class imbalances.

B. Design choices

Considering the SC pipeline design choices, in our experiments we only remove punctuation and lower case all words as prepossessing. For classification in our experiments, we use a bag-of-words approach with unigrams and bigrams and use all features that occur more than 4 times in the training corpus and at maximum in 40% of the training documents. We weight the features according to the TF-IDF weighting scheme. We deliberately do not use word embeddings since the lower dimensional projection needs too much textual domain data to construct. Using data from other domains will distort results as the representation becomes dependent on other domains than the source and target domain that are under performance evaluation. We use LR for its good performance. We use binary classification, $y \in \{0, 1\}$, giving the following loss function for our classifier

$$\ell(x, y)_{LR} = \sum_{i=1}^{n_d} \log\left(1 + e^{-(2y_i-1)x_i\beta+\beta_0}\right) + \alpha\frac{1}{2}\|\beta\|^2$$

where w and w_0 represent the optimizers, α represents the regularization parameter and the hypothesis function is defined as

$$h(x) = \begin{cases} 1 \text{ if } \beta x + \beta_0 > 0 \\ 0 \text{ else.} \end{cases}$$

For α, we use the default classifier settings of our used toolbox [32].

For the Chi2 distance, we use the number of occurrences of a word divided by the occurrences of the top $N = 1000$ words for calculating the probabilities. The $N = 1000$ most occurring words are used to compare means. For the MMD we use a discrete version of (8) [33] where X represents the feature value matrix for the $N = 1000$ most occurring features in the combined set of the two evaluated corpora. We use corpora of the same number of objects, i.e. documents, as input by using a subset of the largest corpus of the two corpora used. For computational convenience we use a maximum of 5000 documents. For EMD, we compare the discrete normalized distributions of the feature count in one document and use the sum of the EMD over the $N = 1000$ most occurring words with available EMD code [34] [35] [36]. Note that the distribution is highly dependent on the length of a document. Therefore, we match document lengths among the two domains. We use the same cluster representatives for both domains, $b = \bar{b} = \{0, 1, \ldots, m\}$ and a distance matrix $D = [d_{i,j}]$ defined by $d_{i,j} = |i - j|$. For the KLD we again use the discrete normalized distributions of the feature count in one document of $\mathbb{P}_x$ and $\bar{\mathbb{P}}_x$ and calculate the KLD as the sum over the $N = 1000$ most occurring features.

We do not compare the distribution of all features for two reasons. In the first place, the distance calculations will take long when using all features. And second, and more important, distributions of rarely occurring features are less reliable.

For the inner domain classification error that is used as predictor in the CMEK model, we use the same logistic regression as for the cross domain classification and use 10-fold cross validation to assess the inner domain classification error.

VII. Results

In this section we will objectively present the results of our experiments according to our performance evaluation described in subsection III-B.

Figure 3 shows the empirical distribution of the relative error as defined in (9) when selecting one source domain. The relative error distributions when randomly selecting a source domain and when using the CMEK source domain selection model are shown.

Table I shows the probability of selecting the true best domain, the probability of selecting one of the five worst domains in terms of the cross domain classification error, and the average cross domain classification error, averaged over the 13 different target domains. We listed the results when individual measures are used for selection, when the linear combination is used (CMEK) and when a source domain is selected at random. The optimal cross domain classification

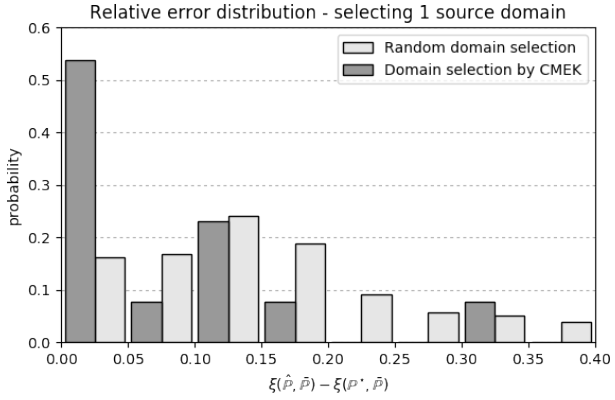


Fig. 3. Performance of CMEK source domain selection model compared to random source domain selection.

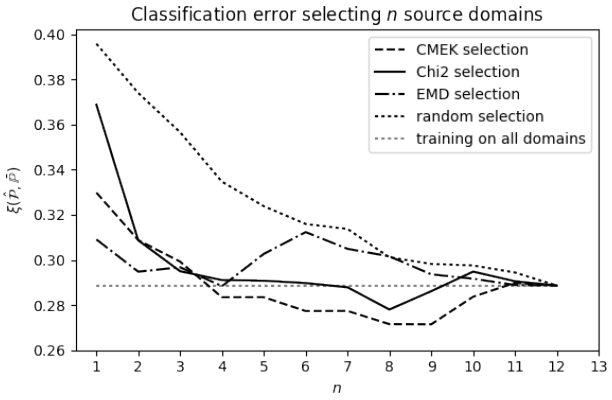


Fig. 4. Performance of CMEK model, Chi2, EMD and random selection for different number of source domains to be selected, compared with training on all source domains.

TABLE I
PERFORMANCE OF CMEK, INDIVIDUAL MEASURES, RANDOM AND OPTIMAL SELECTION

Selection method	**Probability Best possible domain selected**	**Probability One of 5 worst domains selected**	**Average** $\xi(\hat{\mathbb{P}}, \bar{\mathbb{P}})$
Optimal	1	0	.252
EMD	.308	.000	.309
KLD	.308	.077	.324
CMEK	.385	.154	.330
MMD	.077	.154	.350
Chi2	.154	.308	.369
Random	.083	.417	.403

error is listed which can be seen as lower bound of error. The CMEK model uses the optimized weights

$$\hat{\beta} = [0.04,\ 1.19,\ 0.12,\ 0.05,\ 0.66,\ 0.25] \qquad (12)$$

for respectively, Chi2, MMD, EMD, KLD, the source domain inner domain classification error and the constant 1.

Figure 4 shows the cross domain classification error when we let our selection model select n source domains. Results are shown when using CMEK, Chi2, EMD and random selection. Note that the CMEK model is constructed by optimizing the predicted cross domain classification error when training the classifier on single source domains, not training on multiple. The results of using these models for source domain selection are compared with training on all candidate source domains. We choose to present the Chi2 and EMD measures as they show the most interesting behavior over using n domains to train on. The MMD measure has similar behavior as the CMEK model and KLD selection behaves similar to EMD selection.

VIII. CONCLUSION

To select a suitable source domain in terms of cross domain classification error, we hypothesized that this error can be predicted by a function of the source domain distribution $\mathbb{P}$ and marginal target domain distribution $\bar{\mathbb{P}}_x$. We proposed the CMEK model to select the best source domain(s) from a set of candidates. The CMEK model uses a linear combination of the Chi2, MMD, EMD, KLD, the inner domain classification error and the constant 1, with weight vector β as predictor for the cross domain classification error. To evaluate performance, we consider an example including 12 distinct domains forming 132 different source-target domain pairs. The proposed metric parameter β is optimized to minimize the absolute error of the prediction. The optimized predictor was used to select one or multiple best source domains among those 12 domains in order to train a classifier for a thirteenth, unseen target domain. This classifier is tested by calculating the cross domain classification error of the selected source domain(s) and the target domain. The process is repeated 13 times, each time using a different unseen target domain to test on. We benchmark performance with randomly selecting a source domain and with using all candidate source domains to train on.

From figure 3 we see that the CMEK model is well able to identify source domains with low cross domain classification error. In 54% of the runs, the selected source domain was within 5 percent points error of the optimal choice whereas random selection only selected 16% of the times a source domain in this category.

From Table I, we see that compared to random domain selection, the CMEK, MMD, EMD and KLD selection realize significant improvement in average cross domain classification error ($p = 0.0034, p = 0.0054, p = 0.000047$ and $p = 0.0019$ respectively). The CMEK model has a significant larger probability of selecting the best domain compared to random, MMD and Chi2 selection ($p = 0.0029, p = 0.0021$ and $p = 0.038$ respectively). If we look at the probability of not selecting

one of the five worst domains for training, all models except the Chi2 selection model significantly perform better than the random selection model. From (12) we see that the CMEK model depends primarily on the MMD and the inner domain classification error.

Although most selection models significantly improve performance compared to random selecting one source domain, we would still be better off training on all the available source domains. However, if we let our CMEK model select multiple source domains to train on, we are able to get better performance than training on all source domains for some n, see Figure 4. The CMEK model seems to perform significantly better than training on all domains when it selects 8 or 9 domains ($p = 0.047$ and $p = 0.041$). Note, for all mentioned significant results, the normality assumption of the t-test is confirmed, however in this specific evaluation, the pairwise differences seem not normally distributed. What the optimal number of domains to train on is, may be very dependent on what candidate source domains are available. When we have candidate source domains that are somehow similar to the target domain, the optimal n will be higher. For a diverse set of candidate source domains the optimal n will be lower.

Also, from Figure 4, we see that the EMD selection really loses its magic when multiple domains have to be selected. Apparently, the EMD distance is primarily good in detecting the few best domains, and much less capable of estimating the cross domain classification error of medium range source domains. The CMEK model performs significantly better than EMD and random selection in selecting multiple source domains to train on ($p = 0.000023$ and $p = 0.046$ respectively). The CMEK model is for each $3 \geq n \leq 11$ significantly better then random selection. Compared to the CMEK model, EMD selection significantly fails in selecting 6, 7 and 8 source domains.

If we are to make a general recommendation for using one of the evaluated models, in the light of general performance, we would choose the CMEK model. The CMEK model shows significantly good performance and stable behavior in selecting multiple source domains and it has solid performance in selecting the single best domain.

In retrospect to our hypothesis (4), we can conclude that it is to some extend possible to approximate the cross domain classification error as we were able to use this approximation for successful source domain selection. However, there is much more room for improvement.

A. Further discussion

Reflecting on our conclusion, we would like to place some remarks. We showed that the CMEK model works superior in selecting one source domain compared to random selection. However, the results show that, even for candidate source domains with a wide spread in topic and source medium, it is quite beneficial to train on all the candidate source domains. One of the reasons not to train on all data could be that it is too computational expensive. Another reason might be that the candidate source domains are too diverse. To establish if this is the case, we would need a measure that informs us about the diversity of the candidate source domains.

When we know we have candidate source domains that are similar to each other in terms of expressing sentiment, it might very well be that our CMEK model is not able to improve performance compared to training on all data. Therefore, if we have candidate source domains that are similar to each other, we might choose to train on all domains for simplicity.

Another disadvantage of the CMEK model, is that it uses some distance measures that are quite expensive to calculate. This can be a problem when we have many or large candidate source domains. This challenge might be addressed by using less features and more informative features to calculate distance over and we could optimize used code on efficiency. We might investigate what happens to the performance of the CMEK model in case we leave out the Chi2 and KLD measure, as they are given fewest importance, i.e. low weight β_i.

We constructed, tested and evaluated the model with the 13 acquired data sets. We would like to remind that sentiment expression is characterized by a severe notion of stochasticity. It is hard to tell how much performance of the CMEK model will deviate when 13 other sets are chosen to construct the model. Furthermore, we have not evaluated the model when it is constructed on only 3 or 4 data sets for example. For more available domains to construct the model, we might assume that the model will be more refined and better performing due to the wider support for the β weight vector.

Although we described a method that can be easily implemented in other fields of machine learning that encounter a domain shift such as computer vision, fraud detection or spam detection, we only showed our model works reasonably well in the domain of SC.

B. Future Work

As the selection models presented in Table I are to some extension able to select a source domain with lower than average cross domain classification error, we concluded that the measures are to some extend able to predict the cross domain classification error in order to select a suitable source domain. However, if our model would fully do justice to our hypothesis (4), we would improve even further. We could say that proof of concept is given: it is possible to roughly predict the cross domain classification error based on the distribution of the source domain distribution and marginal distribution of the target domain. However, we still see much room for improvement as the lower bound for the error when selecting one domain is .252, we're only half way.

If we would like to improve results further, first steps would be to look closer into the features over which we measure distance. Should we use the 1000 most common, or is distance better measured with more or less features? This could be dependent on corpus size as well. There might be feature selection methods to select the features that are most informative in terms of distance, instead of simply using the N most occurring features.

To improve, we might add some predictors to the linear combination, such as the document length distribution or the length of a corpus. Brief statement are on average more similar with each other than short statements compared with long expressions. Also, if we have more objects in a source domain, we might benefit more when training on that large source domain than training on a corpus with only a few documents. However, with more elements in β, we might need more data to prevent over fitting.

We can also extend our model to give a weight factor on every source domain: source domains that are predicted not suitable get a low weight, domains with low distance to the target are given more weight. This approach leans towards importance sampling.

We constructed a predictor for the cross domain classification error of one source domain, and used the model to select multiple source domains. If this is what we are interested in, it would make sense to calculate distances between a set of multiple domains and the target domain and make a selection model optimized on predicting the cross domain classification error when training on multiple domains. In that case, synergies of data sets will be taken into account. This approach will, however, be very computational expensive.

REFERENCES

[1] V. Hatzivassiloglou and K. R. McKeown, "Predicting the semantic orientation of adjectives," in *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 1997, pp. 174–181.

[2] B. Pang, L. Lee *et al.*, "Opinion mining and sentiment analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[3] L. Terveen, W. Hill *et al.*, "Phoaks: a system for sharing recommendations," *Communication of the ACM*, vol. 40, no. 3, pp. 59–63, 1997.

[4] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of computational science*, vol. 2, no. 1, pp. 1–8, 2011.

[5] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe, "Predicting elections with twitter: What 140 characters reveal about political sentiment." *Icwsm*, vol. 10, no. 1, pp. 178–185, 2010.

[6] S. Tan, X. Cheng, Y. Wang, and H. Xu, "Adapting naive bayes to domain adaptation for sentiment analysis," *Advances in Information Retrieval*, pp. 337–349, 2009.

[7] Y. He, C. Lin, and H. Alani, "Automatically extracting polarity-bearing topics for cross-domain sentiment classification," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 123–131.

[8] J. Blitzer, M. Dredze, F. Pereira *et al.*, "Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification," in *ACL*, vol. 7, 2007, pp. 440–447.

[9] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *Proceedings of the 2006 conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2006, pp. 120–128.

[10] W. M. Kouw, L. J. Van Der Maaten, J. H. Krijthe, and M. Loog, "Feature-level domain adaptation," *Journal of Machine Learning Research*, vol. 17, no. 171, pp. 1–32, 2016.

[11] S. Cohen and L. Guibasm, "The earth mover's distance under transformation sets," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. IEEE, 1999, pp. 1076–1083.

[12] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2960–2967.

[13] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[14] D. Bollegala, "Dynamic feature scaling for online learning of binary classifiers," *Knowledge-Based Systems*, vol. 129, pp. 97–105, 2017.

[15] D. Wu, V. J. Lawhern, and B. J. Lance, "Reducing offline bci calibration effort using weighted adaptation regularization with source domain selection," in *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3209–3216.

[16] K. Vogt, A. Paul, J. Ostermann, F. Rottensteiner, and C. Heipke, "Boosted unsupervised multi-source selection for domain adaptation," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 4, p. 229, 2017.

[17] E. W. Xiang, S. J. Pan, W. Pan, J. Su, and Q. Yang, "Source-selection-free transfer learning," in *IJCAI proceedings-international joint conference on artificial intelligence*, vol. 22, no. 3, 2011, p. 2355.

[18] D. Sutherland, "Two-sample tests, integral probability metrics, and gan objective," 2017.

[19] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[20] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. Mar, pp. 723–773, 2012.

[21] S. Adhikari. (2014) Summary and discussion of: a kernel two-sample test. [Online]. Available: http://www.stat.cmu.edu/~ryantibs/journalclub/mmd.pdf

[22] K. Pearson, "X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, 1900.

[23] R. L. Dobrushin, "Prescribing a system of random variables by conditional distributions," *Theory of Probability & Its Applications*, vol. 15, no. 3, pp. 458–486, 1970.

[24] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[25] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[26] F. H. Khan, U. Qamar, and S. Bashir, "Swims: Semi-supervised subjective feature weighting and intelligent model selection for sentiment analysis," *Knowledge-Based Systems*, vol. 100, pp. 97–111, 2016.

[27] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1188–1196.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[29] D. Kotzias, M. Denil, N. De Freitas, and P. Smyth, "From group to individual labels using deep features," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 597–606.

[30] University of Michigan. (2011) Umich si650 - sentiment classification. [Online]. Available: https://www.kaggle.com/c/si650winter11/data

[31] CrowdFlower. (2017) Data for everyone. [Online]. Available: https://www.crowdflower.com/data-for-everyone/

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[33] V. van Asch. (2012) mmd.py. [Online]. Available: https://www.clips.uantwerpen.be/~vincent/thesis-software/mmd.py

[34] W. Mayner. (2017) pyemd. [Online]. Available: https://github.com/wmayner/pyemd

[35] O. Pele and M. Werman, "A linear time histogram metric for improved sift matching," in *Computer Vision–ECCV 2008*. Springer, October 2008, pp. 495–508.

[36] O. Pele and M. Werman, "Fast and robust earth mover's distances," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, September 2009, pp. 460–467.

Appendix B

Supportive figures and tables

In this chapter, we show results of own experiments. The data set used is not to be made public due to privacy issues. The data set holds over 5000 English reviews of employees all over the world about their shared employer. Performance is reported as classification accuracy or error and experiments are performed in 10-fold cross validation. The classifiers used are, if not specifically mentioned, in default setting from the scikit-learn toolbox [47]. If not specifically mentioned, we used TF-IDF feature weighting with a maximum frequency rate of .8 and minimum frequency of 5. Stopwords are not removed. All words are lower cased and punctuation is removed.

Figure B-1 shows the classification accuracy when using N-grams as features. The accuracies are calculated for nine different on-par performing classifiers. The presented accuracies are averages over a model that selects 1200, 2400 or all features by χ^2 selection. When for a certain N-gram model the classifier is displayed bold, it is the best N-gram model for this classifier.

Figure B-2 shows the classification accuracy when excluding features based on their number of occurrences. Results are calculated with a Ridge Regression classifier and unigrams and bigrams as features.

Figure B-3 shows the increase in classification accuracy when using lemmatization in combination with a unigram + bigram model. The presented accuracy's are averages over a model that selects 1200, 2400 or all features by χ^2 selection.

Figure B-4 shows the classification accuracy when χ^2-selection is used to select features. Two training set sizes are used to see if behavior is subjected to the training set size.

Figure B-5 shows the classification error rate as defined in 3-2 for 9 popular classifiers.

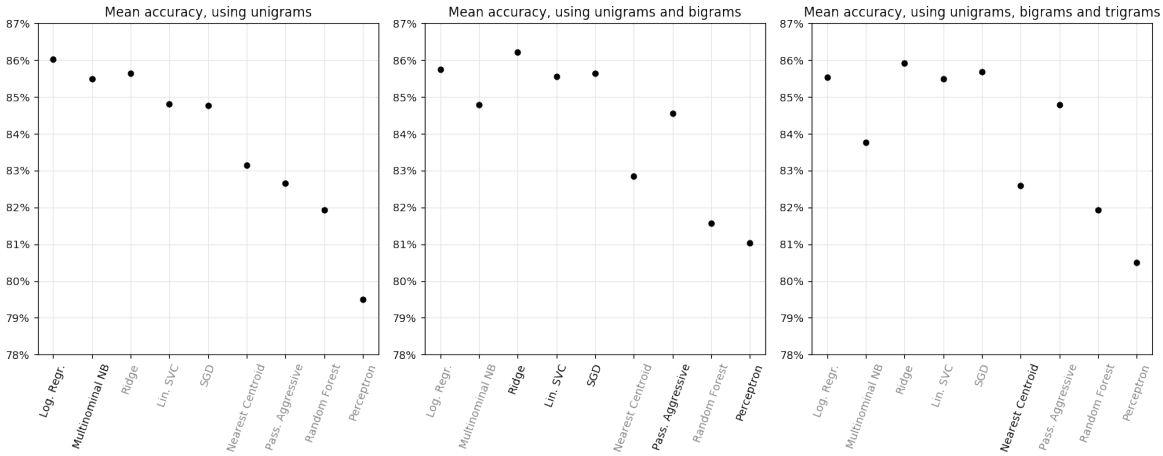


Figure B-1: Performance of using unigrams, unigrams + bigrams, and unigrams + bigrams + trigrams as features.

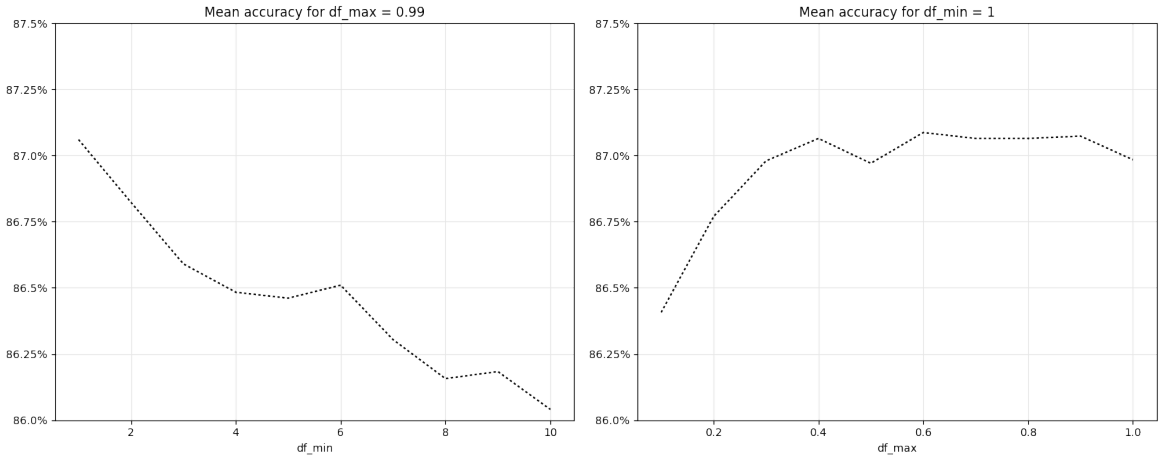


Figure B-2: Performance when excluding features that occur less than n times (left) or in more than ratio r of the documents (right).

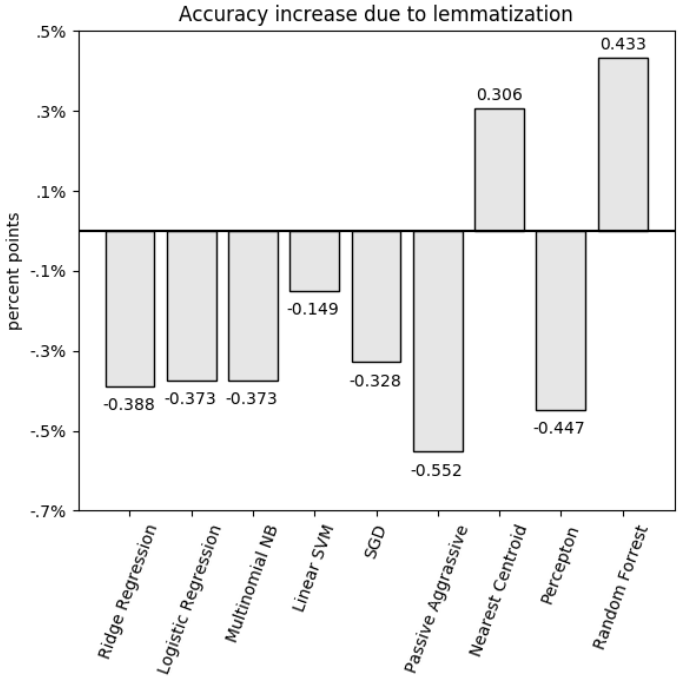


Figure B-3: Classification accuracy increase due to using lemmatization.

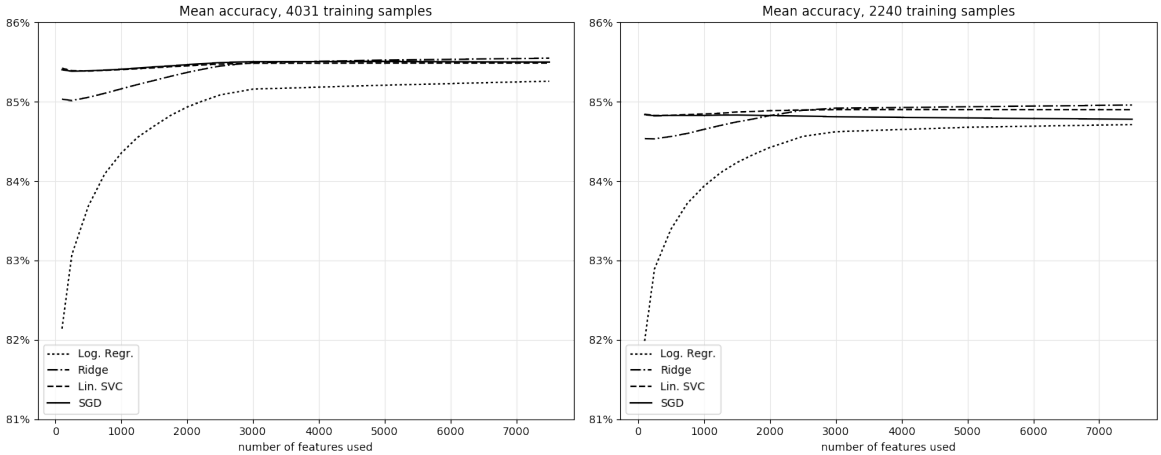


Figure B-4: Feature curves for χ^2-selection

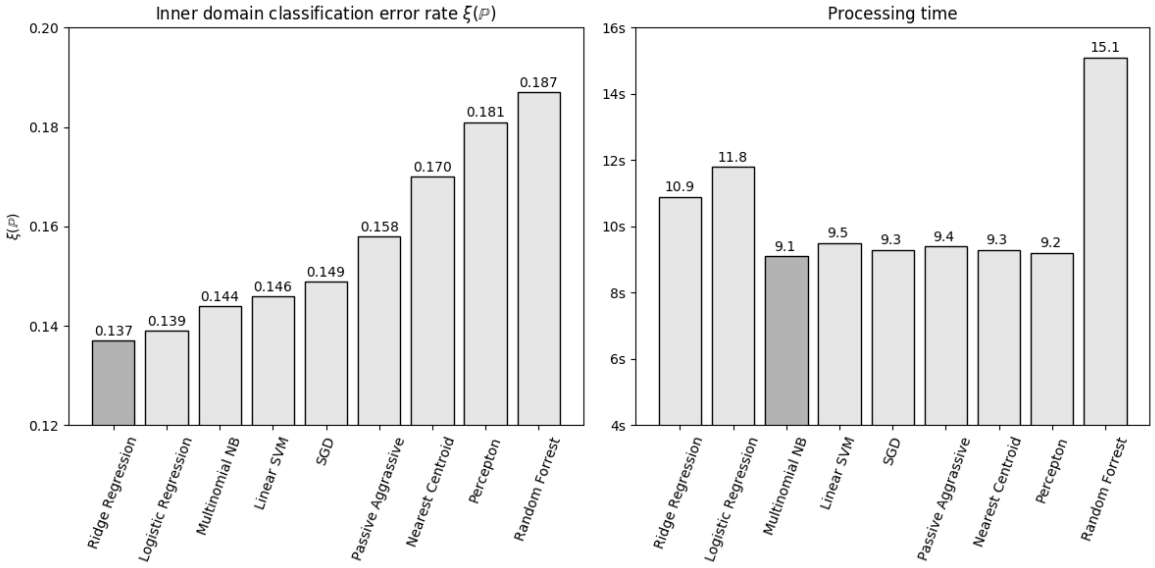


Figure B-5: Inner domain classification performance for 9 on-par performing classifiers.

Table B-1: Adaptation loss for 182 unique source-target domain pairs of our 14 evaluated data sets. The integer numbers correspond to data sets in Table 4-1. The rates are averages over using Ridge Regression, Logistic Regression, Multinomial NB and a Linear SVM.

target source	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0.30	0.14	0.11	0.25	0.09	0.10	0.13	0.17	0.10	0.34	0.24	0.11	0.37
2	0.31	0	0.20	0.21	0.25	0.16	0.15	0.15	0.19	0.29	0.30	0.20	0.19	0.35
3	0.15	0.11	0	0.20	0.20	0.03	0.00	0.00	0.02	0.10	0.19	0.36	0.13	0.34
4	0.19	0.28	0.15	0	0.34	0.12	0.11	0.11	0.17	0.06	0.42	0.16	0.04	0.45
5	0.31	0.38	0.19	0.28	0	0.18	0.16	0.18	0.23	0.39	0.07	0.35	0.25	0.13
6	0.15	0.23	0.14	0.14	0.23	0	0.06	0.09	0.16	0.17	0.40	0.17	0.12	0.40
7	0.14	0.20	0.12	0.16	0.22	0.05	0	0.06	0.14	0.16	0.36	0.22	0.15	0.43
8	0.29	0.31	0.17	0.25	0.29	0.09	0.08	0	0.18	0.25	0.28	0.26	0.19	0.24
9	0.29	0.26	0.13	0.15	0.30	0.13	0.15	0.17	0	0.18	0.41	0.20	0.10	0.41
10	0.12	0.21	0.09	0.08	0.31	0.09	0.07	0.10	0.15	0	0.29	0.26	0.03	0.47
11	0.31	0.29	0.20	0.38	0.12	0.16	0.17	0.17	0.25	0.47	0	0.38	0.34	0.07
12	0.50	0.62	0.29	0.30	0.29	0.33	0.31	0.31	0.35	0.36	0.46	0	0.34	0.42
13	0.22	0.27	0.14	0.11	0.23	0.15	0.14	0.18	0.14	0.10	0.34	0.29	0	0.05
14	0.31	0.29	0.16	0.33	0.22	0.16	0.17	0.16	0.24	0.31	0.15	0.28	0.27	0

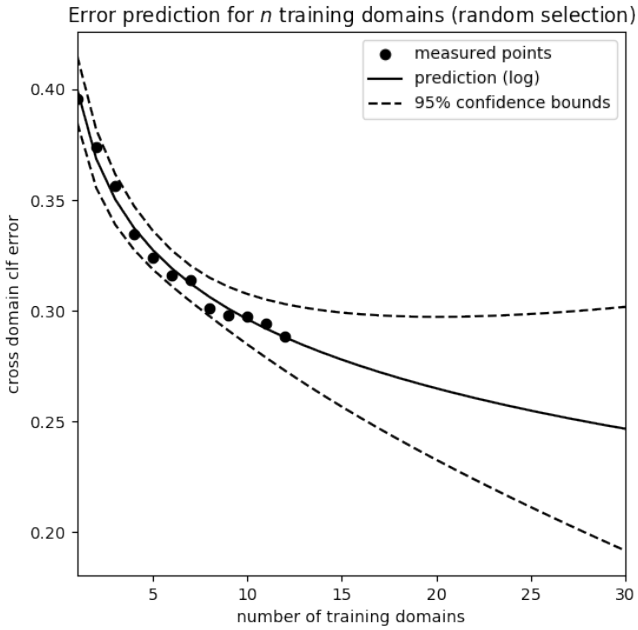


Figure B-6: Prediction and 95% confidence interval of cross domain classification error for training on n domains.

Figure B-6 shows the cross domain classification error rate as defined in 3-2 for training on n randomly selected domains. For the predictions, we fitted a logarithmic model.

The Tables B-2 and B-3 show the probability of M_2 performing better than M_1. For example, looking at Table B-2, we see that there is a very small probability of MMD performing better than CMEK ($p = 0.0021$), we can conclude that the proposed model (CMEK) is significantly better than the MMD model in selecting the single best source domain.

The Tables B-5 to B-9 show the probability that the difference is normally distributed, and the probability of an event of equal or greater extremeness being observed. For example, looking at Table B-4, we see that if we let the CMEK model select 7 source domains to train on ($n = 7$), the difference over the 13 runs was likely to follow a normal distribution $p = 0.86 > 0.05$, we conclude the normality assumption of the paired t-test is not rejected. From the last column, we see that there is a low probability of an observation of equal or greater extremeness, $p = 0.0017$. We conclude the difference is significant.

very small probability of MMD performing better than CMEK ($p = 0.0021$), we can conclude that CMEK is significantly better than the MMD model in selecting the single best source domain.

Table B-2: Probability of M_2 performing equally good or better than M_1 in selecting single best source domain.

		M2					
	Selection model	**Random**	**EMD**	**KLD**	**CMEK**	**MMD**	**Chi2**
M1	**EMD**	0.019	x	0.60	0.80	0.014	0.13
	KLD	0.019	0.60	x	0.80	0.014	0.13
	CMEK	0.0029	0.37	0.37	x	0.0021	0.038
	MMD	0.68	0.99	0.99	1.0	x	0.89
	Chi2	0.30	0.94	0.94	0.98	0.26	x

Table B-3: Probability of M_2 performing equally good or better than M_1 in not selecting one of the 5 worst source domains.

		M2					
	Selection model	**Random**	**EMD**	**KLD**	**CMEK**	**MMD**	**Chi2**
M1	**EMD**	0.000091	x	0.35	0.11	0.11	0.0084
	KLD	0.0093	1.0	x	0.38	0.38	0.057
	CMEK	0.045	1.0	0.93	x	0.68	0.19
	MMD	0.045	1.0	0.93	0.68	x	0.19
	Chi2	0.31	1.0	1.0	0.96	0.96	x

Table B-4: Significance of difference in results of random and CMEK selecting n domains.

	random - CMEK	
n	p normal	p value
2	0.013	<1.0E-3
3	0.12	0.013
4	0.18	<1.0E-3
5	0.75	<1.0E-3
6	0.72	<1.0E-3
7	0.86	0.0017
8	0.08	<1.0E-3
9	0.48	0.0026
10	0.97	0.0036
11	0.85	0.096

Table B-5: Significance of difference in results of random and Chi2 selecting n domains.

	random - Chi2	
n	p normal	p value
2	0.65	<1.0E-3
3	0.44	<1.0E-3
4	0.64	0.0019
5	0.14	0.0046
6	0.47	0.014
7	0.008	0.0030
8	0.0021	0.031
9	0.59	0.050
10	0.67	0.25
11	0.62	0.10

Table B-6: Significance of difference in results of random and EMD selecting n domains.

	random - EMD	
n	p normal	p value
2	0.60	<1.0E-3
3	0.46	<1.0E-3
4	0.11	0.0043
5	0.71	0.083
6	0.10	0.40
7	0.18	0.21
8	<1.0E-3	0.48
9	0.61	0.17
10	0.50	0.10
11	0.65	0.052

Table B-7: Significance of difference in results of CMEK and Chi2 selecting n domains.

	CMEK - Chi2	
n	p normal	p value
2	0.032	0.50
3	0.75	0.31
4	0.10	0.24
5	0.022	0.26
6	0.62	0.10
7	0.48	0.053
8	0.42	0.12
9	<1.0E-3	0.024
10	0.82	0.019
11	0.12	0.43

Table B-8: Significance of difference in results of CMEK and EMD selecting n domains.

	CMEK - EMD	
n	p normal	p value
2	0.32	0.25
3	0.13	0.43
4	<1.0E-3	0.36
5	0.0032	0.13
6	0.15	0.029
7	0.15	0.023
8	0.07	0.0094
9	0.0032	0.014
10	0.45	0.063
11	0.04	0.41

Table B-9: Significance of difference in results of Chi2 and EMD selecting n domains.

	Chi2 - EMD	
n	p normal	p value
2	0.74	0.12
3	0.10	0.43
4	0.35	0.35
5	0.52	0.11
6	0.075	0.081
7	0.059	0.13
8	0.028	0.050
9	0.16	0.12
10	0.51	0.020
11	<1.0E-3	0.094

Appendix C

Examples and derivations

C-1 MI-score example

The following example shows how the MI-score is calculated. Consider the table

Feature f_j	Negative class (0)	Positive class (1)
Absent (0)	4	7
Present (1)	6	3

Table C-1: Occurrence of feature f_j for $N = 20$ documents split by class.

where the values reflect drawings from random variables F and C, we can treat these samples as objects (document and corresponding label). The MI is now calculated with the following formula

$$MI(F;C) = \sum_{e_f \in \{0,1\}} \sum_{e_c \in \{0,1\}} P(F = e_f, C = e_c) log_2 \left(\frac{P(F = e_f, C = e_c)}{P(F = e_f)P(C = e_c)} \right) \tag{C-1}$$

where F is random variable that takes value $e_f = 0$ if the feature is not present and takes the value $e_f = 1$ if the feature is present, C is a random variable taking value $e_c = 0$ if the class is negative and $e_c = 1$ when the class is positive [24]. Note that when presence of the feature f_j denoted by F is not correlated with the class C, the logarithmic function becomes 0 and the MI-score will be 0, indicating that there is no mutual information between F and C. For the given data in Table C-1, we calculate the MI-score according to (C-1).

$$MI(F,C) = \frac{4}{20} log_2\left(\frac{20\times 4}{11\times 10}\right) + \frac{6}{20} log_2\left(\frac{20\times 6}{9\times 10}\right) + \frac{7}{20} log_2\left(\frac{20\times 7}{11\times 10}\right) + \frac{3}{20} log_2\left(\frac{20\times 3}{9\times 10}\right) = 0.07.$$

C-2 Chi squared value example

The following example shows how the χ^2-value is calculated. Consider Table C-1. The expected number of drawings from a mixed class set when performing N drawings in total, are calculated as

$$E_{e_f,e_c} = N \times P(F = e_f) \times P(C = e_c).$$

This gives us the values displayed in C-2 for E_{e_f,e_c}.

Feature f_j	Negative class (0)	Positive class (1)
Absent (0)	5.5	5.5
Present (1)	4.5	4.5

Table C-2: Expected occurrence of feature f_j for $N = 20$ documents split by class assuming independence of F and C.

Now, the χ^2-vale is calculated [24] as

$$\chi^2(F,C) = \sum_{e_f\in\{0,1\}} \sum_{e_c\in\{0,1\}} \frac{\left(N_{e_t e_c} - E_{e_t e_c}\right)^2}{E_{e_t e_c}}$$

with N_{e_f,e_c} in Table C-1 and E_{e_f,e_c} in Table C-2. For our example we calculate

$$\chi^2(F,C) = \frac{(5.5-4)^2}{5.5} + \frac{(5.5-7)^2}{5.5} + \frac{(4.5-6)^2}{4.5} + \frac{(4.5-3)^2}{4.5} = 1.82$$

.

C-3 Maximizing likelihood for normal distribution

This section shows the derivation of maximizing the likelihood for the regular linear regression problem in chapter 3 section 3-4.

Let us define $\hat{\beta}$ as β that maximizes the log-likelihood $\mathcal{L}$. We show that this $\hat{\beta}$ is also the argument that minimizes the least square solution.

$$\begin{aligned}
y &\sim N(\beta x, \sigma^2) \\
\hat{\beta} &= \arg\max_{\beta} \mathcal{L}(x|y) \\
&= \arg\max_{\beta} \prod_{i=1}^{n_{tr}} P_Y(y_i|x, \sigma^2) \\
&= \arg\max_{\beta} \prod_{i=1}^{n_{tr}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i-\beta x)^2}{2\sigma^2}} \\
&= \arg\max_{\beta} \sum_{i=1}^{n_{tr}} \log\left(e^{-\frac{(y_i-\beta x)^2}{2\sigma^2}}\right) \\
&= \arg\min_{\beta} \sum_{i=1}^{n_{tr}} (y - \beta x)^2
\end{aligned}$$

or in matrix form

$$\hat{\beta} = \arg\min_{\beta} (Y - \beta X)^2.$$

This optimization problem has the closed form solution derived as

$$\begin{aligned}
\hat{\beta} &= \arg\min_{\beta} (Y - \beta X)^2 \\
&= \arg\min_{\beta} (Y - \beta X)^T (Y - \beta X) \\
&= \arg\min_{\beta} Y^T Y - 2\beta^T X^T Y + \beta^T X^T X \beta
\end{aligned}$$

taking the partial derivative with respect to β and finding the minimum by setting this derivative equal to zero gives

$$\begin{aligned}
\frac{\partial}{\partial \hat{\beta}} \left(Y^T Y - 2\hat{\beta}^T X^T Y + \hat{\beta}^T X^T X \hat{\beta}\right) &= 0 \\
-2X^T Y + 2X^T X\hat{\beta} &= 0 \\
\hat{\beta} &= (X^T X)^{-1} X^T Y.
\end{aligned}$$

We now have a hypothesis function $\hat{h} : \mathbb{X} \to \mathbb{Y}$ defined as

$$\hat{h} : y_i = \begin{cases} 1 & x_i\hat{\beta} > 0 \\ 0 & \text{else.} \end{cases}$$

C-4 Maximizing likelihood for logistic distribution

This section shows the derivation of maximizing the likelihood for the logistic regression problem in chapter 3 section 3-4.

Let us define $\hat{\beta}$ as β that minimizes the loss function.

$$
\begin{aligned}
\hat{\beta} &= \arg\max_{\beta} \mathcal{L}(x|y) \\
&= \arg\max_{\beta} \prod_{i=1}^{n_{tr}} P_Y(y_i = 1|x_i)^{y_i} P_Y(y_i = 0|x_i)^{1-y_i} \\
&= \arg\max_{\beta} \sum_{i=1}^{n_{tr}} y_i \log\left(P_Y(y_i = 1|x_i)\right) + (1 - y_i) \log\left(P_Y(y_i = 0|x_i)\right) \\
&= \arg\max_{\beta} \sum_{i=1}^{n_{tr}} y_i \log\left(\frac{1}{1 + e^{-\beta x}}\right) + (1 - y_i) \log\left(\frac{1}{1 + e^{\beta x}}\right) \\
&= \arg\max_{\beta} \sum_{i=1}^{n_{tr}} y_i \beta x - \log\left(1 + e^{\beta x}\right)
\end{aligned}
$$

Appendix D

More on word embeddings

In this chapter, we will take a closer look at the properties of embedded feature spaces, how to construct them and what variations are popular.

D-1 Properties

The embedded feature space constructed by popular models [36] [14], has the property that similar words have similar vector representations in the lower dimensional embedded feature space. Assume we have two words w_i and w_j which have similar meaning, they are synonyms represented in the space $\mathbb{X} \in \mathbb{R}^{|\mathcal{F}| \gg m}$. We apply a mapping $C : \mathbb{X} \rightarrow \mathbb{X}_{emb} \in \mathbb{R}^m$ that maps w_i and w_j into the m-dimensional space giving $C(w_i) = v_i$ and $C(w_j) = v_j$ where $v_i \in \mathbb{R}^m$ and $v_j \in \mathbb{R}^m$. Now, since w_i and w_j have similar meaning, we construct C such that $||v_i - v_j|| \approx 0$. In practise the mapping $w_i \rightarrow v_i$ is performed by a table look up. Assume we choose all unique words as features, $\mathcal{F} = W$. In this case, C is a matrix of size $|W| \times m$ where $|W|$ is the number of unique words in the corpus, the length of the vocabulary W.

Another very interesting property of this embedded feature space for NLP, is that the similarity of the vector representations of words is not limited to syntactic properties alone. These representations are surprisingly good at capturing semantic regularities in language as well[72]. It is possible to perform vector operations to add or subtract certain semantic and syntactic properties of a word, see Figure D-1. Below a few examples of vector operations showing that the embedding by C captures semantic notions [14],

1. $C(\text{“king”}) - C(\text{“man”}) + C(\text{“woman”}) \approx C(\text{“queen”})$
2. $C(\text{“Paris”}) - C(\text{“France”}) + C(\text{“Germany”}) \approx C(\text{“Berlin”})$
3. $C(\text{“biggest”}) - C(\text{“big”}) + C(\text{“small”}) \approx C(\text{“smallest”})$.

It appears that this embedded feature space captures semantic notions such as capital cities, currencies and man-woman equivalence but also syntactic relations such as opposites, comparatives and grammatical conjugates. This suggests that it might be possible that also notions such as objective and subjective or positive and negative are included in the space. In this case, it should be possible to train a classifier that distinguishes a positive sentiment subspace and a negative sentiment subspace. And indeed, research has proven that this embedded feature space can be used for effective SC [73].

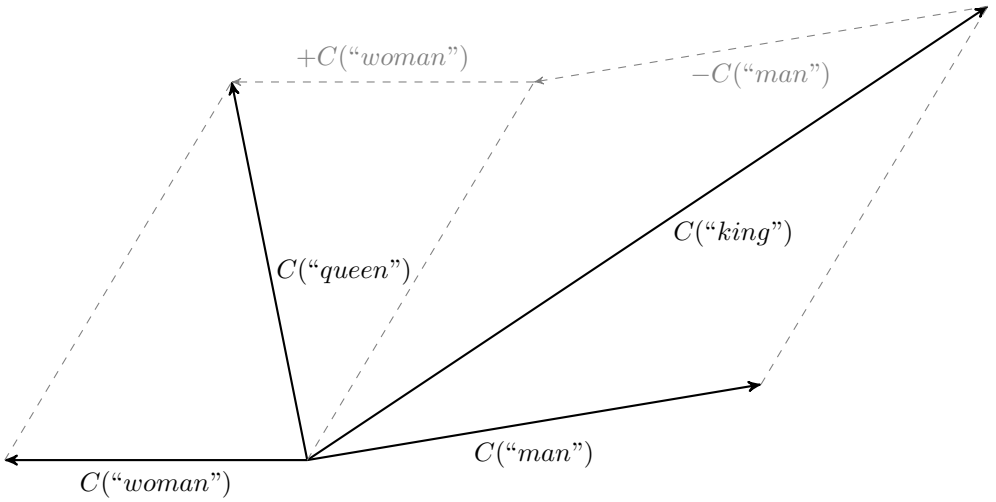


Figure D-1: Illustrative example of a vector operation in the embedded feature space.

D-2 Model

The models that construct an embedded feature space are based on the assumption that words with similar meanings occur in similar context. Let the m-dimensional feature space be notated as $\mathbb{X}_{emb}$ and $C(w_i)$ being the mapping of a word w_i to $v_i \in \mathbb{X}_{emb}$. Now, the context of the word w_i is defined as the N words subsequent to w_i denoted as w_{i-N}^{i-1}. The assumption can be formulated as

$$\begin{array}{ll} \text{IF} & P(w_i|w_{i-N}^{i-1}) \approx P(w_j|w_{j-N}^{j-1}) \quad \text{AND} \quad w_{i-N}^{i-1} = w_{j-N}^{j-1} \\ \text{THAN} & C(w_i) \approx C(w_j). \end{array}$$

A simplification of Bengio's model assuming no output or hidden layer bias, uses a look up table C to transform the N-context words into N feature vectors in the m-dimensional embedded space. These vectors are used as input for a neural network with one layer and h hidden units. The h-dimensional output of the neural network is mapped to an output vector y with $|W|$ elements by a function U. A softmax function is used to add the constraint that all conditional probabilities add up to 1 and are non-negative. See Fig. D-2 for an schematic representation of the model.

The model consists of $N \times m + N \times m \times h + |W| \times h = (h+1) \times n \times m + |W| \times h$ parameters to be estimated per word w_i where $|W|$ is clearly the most influential factor for dimensionality. It estimates a conditional probability of w_i by giving its context w_{i-N}^{i-1} as input. The sum

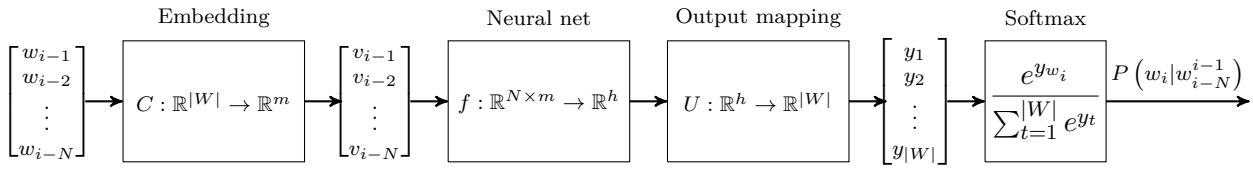


Figure D-2: Schematic representation of a simplified version of Bengio's model.

of all log-likelihoods of all estimated conditional probabilities for $i \in I$ has to be maximized, where I is the total number of words in the corpus. With $\mathcal{L}$ being the likelihood function, we have the optimization problem

$$\max_{C,f,U} \quad \sum_{i=1}^{I} \log \mathcal{L}\left(P\left(w_i \middle| w_{j-N}^{j-1}\right) \middle| \text{corpus}\right).$$

With the assumption that similar words occur in similar contexts in mind, we know that when the conditional probability function of two different contexts is more or less equal, the representations of the context words in $\mathbb{X}_{emb}$ must be more or less equal. The model forces this to be true. An example, let us have two sentences with different context with similar meaning

$$\begin{array}{cccccc} \underset{w_1}{\text{the}} & \underset{w_2}{\text{cat}} & \underset{w_3}{\text{walks}} & \underset{w_4}{\text{in}} & \underset{w_1}{\text{the}} & \underset{w_5}{\text{room}} \\ \underset{w_1}{\text{the}} & \underset{w_6}{\text{dog}} & \underset{w_3}{\text{walks}} & \underset{w_4}{\text{in}} & \underset{w_1}{\text{the}} & \underset{w_5}{\text{room}} \end{array}$$

The proposed model calculates the conditional probability distribution of w_5

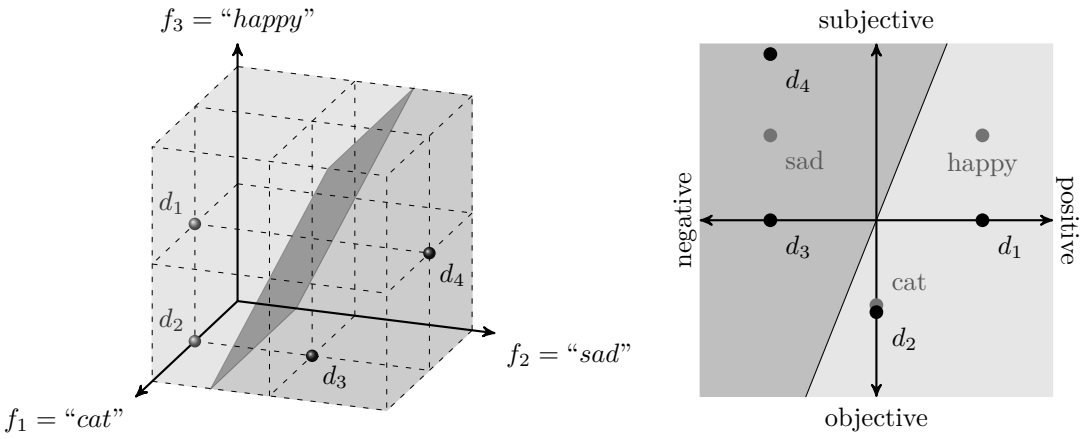
$$\begin{aligned} P(w_5|w_1, w_2, w_3, w_4, w_1) &= U\left(f\left(C(w_1), C(w_2), C(w_3), C(w_4), C(w_1)\right)\right) \\ P(w_5|w_1, w_6, w_3, w_4, w_1) &= U\left(f\left(C(w_1), C(w_6), C(w_3), C(w_4), C(w_1)\right)\right). \end{aligned}$$

Since the likeliness of the word *room* to occur subsequent to both contexts is equal, we have equal probability distribution functions for both contexts.

$$\begin{aligned} P(w_5|w_1, w_2, w_3, w_4, w_1) &= P(w_5|w_1, w_6, w_3, w_4, w_1) \\ U\left(f\left(C(w_1), C(w_2), C(w_3), C(w_4), C(w_1)\right)\right) &= U\left(f\left(C(w_1), C(w_6), C(w_3), C(w_4), C(w_1)\right)\right) \end{aligned}$$

imposing a C such that

$$\begin{aligned} ||C(w_2) - C(w_6)|| &\approx 0 \\ ||C(\text{“cat”}) - C(\text{“dog”})|| &\approx 0. \end{aligned}$$



(a) Raw term frequency feature space.

(b) Embedded feature space sketch.

Figure D-3: Feature space transformation from raw term frequency to embedded space.

D-3 Model variations

Newer versions of Bengio's model are designed to improve computational speed, semantic accuracy and syntactic accuracy. One of the most popular algorithms is the *word2vec* algorithm developed by Mikolov [14]. This algorithm is based on a simplified version of Bengio's model. The hidden layer of the neural network is removed from the model as shown in Figure D-2 and a hierarchical softmax function is used. Both changes reduce the number of parameters to be estimated drastically. This model is called the Continuous Bag-of-Words (CBOW) model because the order of words is not taken into account (bag-of-words) and it uses a continuous distributed representation of the context. The number of parameters to be estimated per w_i are in this model $N \times m + m \times log_2(|W|)$.

Another model of Mikolov is called the Continuous Skip-gram Model. For this model, instead of predicting the current word based on the context, it tries to maximize classification of a word based on another word in the same sentence. When the maximum distance between the context word and target word is called D, the number of parameters to be estimated per w_i is $D \times (m + m \times log_2(|W|)$. This model is thus more computational expensive than the CBOW model but wins in semantic accuracy [14]. Similar approaches can be used to convert a document or paragraph of variable length into a fixed length feature vector. A popular model is from Le and is called *doc2vec*. Using the word2vec algorithm in order to build a feature space used for SC beats the previously best models [39].

If we would use a word embedded feature space for classification of the sample data set from Figure 3-3, we could visualize this as in Figure D-3. Note that this is only a sketch and in used applications, the dimensions of this feature space do not directly represent notions such as objectivity and subjectivity or positive and negative.

Bibliography

[1] TU Delft, "Mission statement," 2017.

[2] TU Delft, "DCSC: Research by Topics," 2017.

[3] TU Delft, "Pattern Recognition Laboratory: Description," 2017.

[4] TU Delft, "Systems & Control graduation guide," 2017.

[5] B. Pang, L. Lee, *et al.*, "Opinion mining and sentiment analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[6] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter, "Phoaks: A system for sharing recommendations," *Communications of the ACM*, vol. 40, no. 3, pp. 59–62, 1997.

[7] Accern, "Our desire to automate the discovery of early news," 2017.

[8] Zendesk, "The impact of customer service on customer lifetime value," 2017.

[9] Mintel, "Online shopping us 2015 report," 2015.

[10] Perez, S., "Analysis of social media did a better job at predicting trump's win than the polls," 2016.

[11] V. Hatzivassiloglou and K. R. McKeown, "Predicting the semantic orientation of adjectives," in *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pp. 174–181, Association for Computational Linguistics, 1997.

[12] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of EMNLP*, pp. 79–86, Association for Computational Linguistics, 2002.

[13] P. D. Turney, "Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 417–424, Association for Computational Linguistics, 2002.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[15] M. Sun, Y. Liu, Z. Liu, and M. Zhang, *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*. Springer, 2015.

[16] C. Van Rijsbergen, *Information Retrieval*. Butterworth-Heinemann, 1979.

[17] Z. Jianqiang and G. Xiaolin, "Comparison research on text pre-processing methods on twitter sentiment analysis," *IEEE Access*, vol. 5, pp. 2870–2879, 2017.

[18] S. Kiritchenko, X. Zhu, and S. M. Mohammad, "Sentiment analysis of short informal texts," *Journal of Artificial Intelligence Research*, vol. 50, pp. 723–762, 2014.

[19] A. Bifet and E. Frank, "Sentiment knowledge discovery in twitter streaming data," in *International conference on discovery science*, pp. 1–15, Springer, 2010.

[20] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," in *Proceedings of the 12th international conference on World Wide Web*, pp. 519–528, ACM, 2003.

[21] A. Tripathy, A. Agrawal, and S. K. Rath, "Classification of sentiment reviews using n-gram machine learning approach," *Expert Systems with Applications*, vol. 57, pp. 117–126, 2016.

[22] J.-C. Na, C. Khoo, and P. H. J. Wu, "Use of negation phrases in automatic sentiment classification of product reviews," *Library Collections, Acquisitions, and Technical Services*, vol. 29, no. 2, pp. 180–191, 2005.

[23] M. Joshi and C. Penstein-Rosé, "Generalizing dependency features for opinion mining," in *Proceedings of the ACL-IJCNLP 2009 conference short papers*, pp. 313–316, Association for Computational Linguistics, 2009.

[24] C. Manning, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[25] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[26] W. B. Frakes, "Term conflation for information retrieval," in *Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 383–389, British Computer Society, 1984.

[27] M. Popovic and P. Willett, "The effectiveness of stemming for natural-language access to slovene textual data," *Journal of the American Society for Information Science*, vol. 43, no. 5, p. 384, 1992.

[28] R. Baeza-Yates, B. Ribeiro-Neto, *et al.*, *Modern information retrieval*, vol. 463. ACM press New York, 1999.

[29] Y. Bao, C. Quan, L. Wang, and F. Ren, "The role of pre-processing in twitter sentiment analysis," in *International Conference on Intelligent Computing*, pp. 615–624, Springer, 2014.

[30] E. Clark and K. Araki, "Text normalization in social media: progress, problems and applications for a pre-processing system of casual english," *Procedia-Social and Behavioral Sciences*, vol. 27, pp. 2–11, 2011.

[31] P.-W. Liang and B.-R. Dai, "Opinion mining on social media data," in *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, vol. 2, pp. 91–96, IEEE, 2013.

[32] P. Shahana and B. Omman, "Evaluation of features on sentimental analysis," *Procedia Computer Science*, vol. 46, pp. 1585–1592, 2015.

[33] G. Paltoglou and M. Thelwall, "A study of information retrieval weighting schemes for sentiment analysis," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, (Stroudsburg, PA, USA), pp. 1386–1395, Association for Computational Linguistics, 2010.

[34] A. McCallum, K. Nigam, *et al.*, "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752, pp. 41–48, Madison, WI, 1998.

[35] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "paper recommender systems: a literature survey," *International Journal on Digital Libraries*, vol. 17, no. 4, pp. 305–338, 2016.

[36] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[37] Schwartz, V., "Representing words," 2016.

[38] B. Xue, C. Fu, and Z. Shaobin, "A study on sentiment computing and classification of sina weibo with word2vec," in *Big Data (BigData Congress), 2014 IEEE International Congress on*, pp. 358–363, IEEE, 2014.

[39] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1188–1196, 2014.

[40] J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the poor assumptions of naive bayes text classifiers," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 616–623, 2003.

[41] P. Kalaivani and K. Shunmuganathan, "Sentiment classification of movie reviews by supervised machine learning approaches," *Indian Journal of Computer Science and Engineering*, vol. 4, no. 4, pp. 285–292, 2013.

[42] V. N. Vapnik, "1982," *Estimation of dependencies based on empirical data*, 1979.

[43] Sewell, M., "Svm's history."

[44] W. Kouw, "Lecture on covariate shift," 2017.

[45] S. Sekine, "The domain dependence of parsing," in *Proceedings of the fifth conference on Applied natural language processing*, pp. 96–102, Association for Computational Linguistics, 1997.

[46] S. Tan, X. Cheng, Y. Wang, and H. Xu, "Adapting naive bayes to domain adaptation for sentiment analysis," *Advances in Information Retrieval*, pp. 337–349, 2009.

[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[48] Crunchr BV., "Employee survey response data set," 2017.

[49] University of Michigan, "Umich si650 - sentiment classification," 2011.

[50] CrowdFlower, "Data for everyone," 2017.

[51] D. Kotzias, M. Denil, N. De Freitas, and P. Smyth, "From group to individual labels using deep features," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 597–606, ACM, 2015.

[52] A. Gretton, A. J. Smola, J. Huang, M. Schmittfull, K. M. Borgwardt, and B. Schölkopf, "Covariate shift by kernel mean matching," 2009.

[53] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *Proceedings of the 2006 conference on empirical methods in natural language processing*, pp. 120–128, Association for Computational Linguistics, 2006.

[54] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[55] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, pp. 2960–2967, IEEE, 2013.

[56] A. Arnold, R. Nallapati, and W. W. Cohen, "A comparative study of methods for transductive transfer learning," in *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pp. 77–82, IEEE, 2007.

[57] D. Wu, V. J. Lawhern, and B. J. Lance, "Reducing offline bci calibration effort using weighted adaptation regularization with source domain selection," in *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, pp. 3209–3216, IEEE, 2015.

[58] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification," in *ACL*, vol. 7, pp. 440–447, 2007.

[59] A. Axelrod, X. He, and J. Gao, "Domain adaptation via pseudo in-domain data selection," in *Proceedings of the conference on empirical methods in natural language processing*, pp. 355–362, Association for Computational Linguistics, 2011.

[60] D. Sutherland, "Two-sample tests, integral probability metrics, and gan objective," 2017.

[61] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[62] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. Mar, pp. 723–773, 2012.

[63] S. Adhikari, "Summary and discussion of: "a kernel two-sample test"," 2014.

[64] K. Pearson, "X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, 1900.

[65] R. L. Dobrushin, "Prescribing a system of random variables by conditional distributions," *Theory of Probability & Its Applications*, vol. 15, no. 3, pp. 458–486, 1970.

[66] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[67] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[68] V. van Asch, "mmd.py," 2012.

[69] W. Mayner, "pyemd," 2017.

[70] O. Pele and M. Werman, "A linear time histogram metric for improved sift matching," in *Computer Vision–ECCV 2008*, pp. 495–508, Springer, October 2008.

[71] O. Pele and M. Werman, "Fast and robust earth mover's distances," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 460–467, IEEE, September 2009.

[72] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations.," in *hlt-Naacl*, vol. 13, pp. 746–751, 2013.

[73] Keydana, S., "Sentiment analysis of moviereviews (3):doc2vec," 2016.

Glossary

List of Acronyms

CBOW	Continuous Bag-of-Words
Chi2	Chi squared
CMEK	Acronym of Chi2, MMD, EMD and KLD
DCSC	Delft Center for Systems and Control
EMD	Earth Mover's Distance
IDF	Inverse Document Frequency
IPMs	Integral Probability Metrics
KLD	Kullback-Leibler Divergence
LR	Logistic Regression
MI	Mutual Information
MMD	Maximum Mean Discrepancy
NB	Naive Bayes
NLP	Natural Language Processing
POS	Part of Speech
PRLab	Pattern Recognition Laboratory
RR	Ridge Regression
SA	Sentiment Analysis
SC	Sentiment Classification
SCL	Structural Correspondence Learning

SVM	Support Vector Machine
TCA	Transfer Component Analysis
TF-IDF	Term Frequency-Inverse Document Frequency

List of Symbols

$\bar{\mathbb{P}}$	Target domain data or distribution
$\bar{\mathbb{P}}_x$	Marginal distribution of the target domain, unlabeled target data
β	Weight vector
β_i	Element in β
χ^2	Pearson's cumulative test statistic
δ	The difference vector between 2 result vectors, or Dirac delta distribution
ℓ	Loss function
ϵ	Stochastic error
Γ	Regularization strength parameter
$\hat{\beta}$	Optimal weight vector
$\hat{\mathbb{P}}$	Predicted best source domain
$\hat{d}$	Optimal distance function
$\hat{h}$	Optimal hypothesis function $\in \mathcal{H}$
$\hat{y}$	Set of predicted class labels
$\hat{y}_i$	Predicted label of d_i by $\hat{h}$
$\mathbb{E}^{\mathbb{P}}[X]$	Expected value of X with $X \sim \mathbb{P}$
$\mathbb{N}$	Set of natural numbers including 0
$\mathbb{P}$	Source domain data or distribution
$\mathbb{P}^\star$	True best source domain
$\mathbb{P}_x$	Marginal distribution of the source domain, unlabeled source data
$\mathbb{R}_+$	Set of positive real numbers including zero
$\mathbb{X}$	Discrete feature space
$\mathbb{X}_{emb}$	Embedded feature space
$\mathbb{Y}$	Discrete class label space
$\mathcal{A}(\mathbb{P}, \bar{\mathbb{P}})$	The $\mathcal{A}$-distance between $\mathbb{P}$ and $\bar{\mathbb{P}}$
$\mathcal{D}$	Family of distance functions
$\mathcal{F}'$	Selected feature set, subset of $\mathcal{F}$
$\mathcal{F}$	Feature set, or set of mapping functions
$\mathcal{H}$	Hypothesis space, or Reproducing Kernel Hilbert space
$\mathcal{L}$	Log-likelihood
$\mathcal{P}$	Set of candidate source domains
$\mathcal{P}'$	Set $\mathcal{P}$ without $\tilde{\mathbb{P}}$

$\tilde{\mathbb{P}}$	Proxy target domain, borrowed from $\mathcal{P}$
$\tilde{\mathbb{P}}_x$	Proxy target domain marginal on $\mathbb{X}$, borrowed from $\mathcal{P}$
$\wp(W)$	Power set of W
$\xi(\mathbb{P})$	Inner domain classification error of domain $\mathbb{P}$
$\xi(\mathbb{P}, \bar{\mathbb{P}})$	Cross domain classification error of a classifier trained on $\mathbb{P}$ and tested on $\bar{\mathbb{P}}$
$\xi_{relative}(\hat{\mathbb{P}}, \bar{\mathbb{P}})$	Relative cross domain classification error
$a(\mathbb{P}, \bar{\mathbb{P}})$	Adaptation loss of a classifier trained on $\mathbb{P}$ and applied on $\bar{\mathbb{P}}$
b	Vector of cluster representatives
C	Random variable in $\{0, 1\}$ that denotes the class
c_x	Unlabeled corpus
D	Distance matrix
d	Distance function $d : \mathbb{P} \times \bar{\mathbb{P}}_x \to \mathbb{R}_+$
d_i	A document or opinion expression
$d_{i,j}$	Element in D
D_{tr}	Set of training documents
D_{ts}	Set of test documents
e_c	Variable that denotes the class
e_f	Variable that denotes presence or absence of feature
E_{e_f,e_c}	Expected number of drawings with properties e_f and e_c
F	Flow Matrix, or random variable in $\{0, 1\}$ that denotes presence of a feature
f	Comparison function for IPM
f_j	The jth feature of $\mathcal{F}$
$f_{i,j}$	Element in flow matrix F
h	A hypothesis function that maps $\mathbb{X} \to \mathbb{Y}$
$h^\star$	True mapping $\mathbb{X} \to \mathbb{Y}$
$h^\star_{\mathbb{P}}$	True hypothesis function for domain $\mathbb{P}$
M	A selection model
n_c	Number of data sets in $\mathcal{P}$
n_F	Number of features
n_s	Number of documents in domain sample
N_{e_f,e_c}	Observed number of drawings with properties e_f and e_c
$n_{F'}$	Number of feature in the selected feature set $\mathcal{F}'$
n_{f_j}	Number of documents in which f_j occurs
n_{tr}	Number of training objects
n_{ts}	Number of test objects
$p(w)$	Probability of w
S	Feature space
s	Vector containing weighted distance functions, a constant and the inner domain classification error
S_+	Subspace of feature space for positive class
S_-	Subspace of feature space for negative class

s_i	Element in s
tf_{ij}	The frequency of term f_j in d_i
U	Output mapping from the neural net output to a class
W	Vocabulary
X	Feature space, or random variable representing a document
x_i	Feature vector of d_i
x_{ij}	The feature value of d_i for feature f_j, element of X
X_{tr}	Feature values matrix of D_{tr}
X_{ts}	Feature values matrix of D_{ts}
Y	Vector of class labels, or random variable representing the class label
y	Set of class labels
y_i	Class label of d_i
Y_{tr}	Labels of the training documents
Y_{ts}	Labels of the test documents
Z	Count vectorizer