TakeLab

Laboratorij za analizu teksta i inženjerstvo znanja
Text Analysis and Knowledge Engineering Lab
Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
Unska 3, 10000 Zagreb, Hrvatska

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS num. 1531

Domain Adaptation for Sentiment Analysis from Text

Matej Paradžik

Zagreb, July 2017.

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING
MASTER THESIS COMMITTEE

Zagreb, 3 March 2017

MASTER THESIS ASSIGNMENT No. 1531

Student: **Matej Paradžik (0036465990)**
Study: Computing
Profile: Computer Science

Title: **Domain Adaptation for Sentiment Analysis from Text**

Description:

The increase in online communication is paralleled by an increase in the amount of user-generated text. Texts cover diverse genres and domains, often expressing users' opinions or sentiments toward various topics, persons, products, etc. However, sentiment analysis systems are typically tailored for a specific domain, which impedes their use on texts from other domains. Besides domain dependence, sentiment analysis systems are often also dependent on a specific text genre, such as microblogs, reviews, or short comments. Developing sentiment analysis systems for each domain and genre separately is time consuming and costly. The problem can be alleviated by the use of domain adaptation techniques.

The focus of this thesis are the domain adaptation techniques for sentiment analysis in Croatian language. Study the domain adaptation techniques, with a focus on domain adaptation for sentiment analysis. Analyze and implement at least three domain adaptation techniques. Apply the techniques to suitable datasets in Croatian and English, covering different domains and genres. Carry out a detailed experimental evaluation of the sentiment analysis system and the domain adaptation techniques, including error analysis and statistical significance testing. All references must be cited, and all source code, documentation, executables, and datasets must be provided with the thesis.

Issue date: 10 March 2017
Submission date: 29 June 2017

Mentor:

Associate Professor Jan Šnajder, PhD

Committee Chair:

Full Professor Siniša Srbljić, PhD

Committee Secretary:

Assistant Professor Tomislav Hrkać, PhD

CONTENTS

1. Introduction

More often than not, individuals ask for attitudes, opinions, and recommendations from friends and family about things such as which smartphone to buy or which restaurant to go out to. Similarly, businesses as well as politicians conduct surveys to find about public opinion on their products, services, or policies. Web fundamentally changed the way people share and acquire information with social media websites like Facebook or microblogging platform Twitter enabling users to share all kinds of information. Most of the online stores have review sections where customers can give their opinions about products they bought. News websites have comment sections where people can discuss current issues. Instead of only relying on word-of-mouth communication, individuals now turn to Web for opinions and recommendations. Businesses and politicians no longer have to pay for conducting surveys, since all needed information is readily available on the Web. However, the number of Internet users went from 400 millions in 2000, which made around 7% of world population, to 46% of world population or 3.4 billions of people in 2016.[1] Consequently, the amount of user-generated text became huge and with it the need for automatic analysis has arisen.

Large amount of user-generated text is opinionated, so sentiment analysis became an important research topic of natural language processing due to its potential applications. Sentiment analysis is a computational study of opinions, sentiments, and attitudes expressed in text towards entities and their attributes (Pang and Lee, 2008). One of the more popular tasks of sentiment analysis is sentiment classification, which is about predicting overall sentiment towards the entity that the given text is about, e.g., classifying reviews as positive or negative. The prevalent approach to tackle sentiment classification is to use machine learning algorithms, specifically supervised learning algorithms (Liu and Zhang, 2012). Supervised machine learning algorithms use labeled training data in order to learn a classifier that tries to predict labels of unseen examples. Such classifiers can perform well enough when used on examples drawn from the same distribution as the training set, but, when that's not the case, their performance can significantly deteriorate. Sentiment classification systems are usually tailored for a certain domain or genre of text, so training set for learning a classifier is compiled from labeled documents from the given target domain and genre. However, user-

[1] `http://www.internetlivestats.com/`; Accessed: 22 April 2017

generated texts come from many different domains, and genres, and classifier learned using training data from the initial target domain would not perform as good on the new domain. To tackle this, one might compile training set for the new target domain and learn the classifier again. Although gathering training data is relatively easy, labeling it can be expensive and tedious as it often requires human effort. Domain adaptation is a field of machine learning which tries to leverage a fact that unlabeled data is easy to come by. Standard domain adaptation scenario is that one wants to learn a classifier for a certain domain, i.e., target domain, for which only unlabeled data is available, using labeled data from different but similar domain, i.e., source domain. For example, one has a corpus of labeled and unlabeled car reviews and wants to learn a classifier that predicts the sentiment of hotel reviews for which only an unlabeled set is available. Domain adaptation methods use both the labeled set from source domain and unlabeled data from the target domain to learn a classifier which performs well on the target domain.

The topic of this thesis are domain adaptation techniques applied on sentiment classification in Croatian language. Sentiment classification based on supervised machine learning is reviewed and reasons for performance degradation in case of cross-domain sentiment classification are discussed. Domain adaptation techniques based on instance weighting and change of representation are studied. Goal of this thesis is to implement and evaluate performance of different domain adaptation techniques for sentiment classification in Croatian and English languages. To this end, linear support vector machine is used as a sentiment classifier and domain adaptation is performed using structural correspondence learning (Blitzer et al., 2007), frustratingly easy domain adaptation (Daumé III, 2007), marginalized stacked denoising autoencoders (Chen et al., 2012), and domain adversarial neural networks (Ajakan et al., 2014). Domain adaptation techniques for sentiment classification are experimentally evaluated on Croatian and English datasets. For English dataset, a standard sentiment domain adaptation dataset of Blitzer et al. (2007) is used, which consists of reviews from four domains: books, DVD, electronics, and kitchen. The Croatian dataset consists of two domains: restaurant reviews and social network comments from telecommunications domain. Performances of used domain adaptation techniques are compared and results are discussed.

The remainder of the thesis is structured as follows. The second chapter reviews area of sentiment analysis with a focus on sentiment classification based on supervised machine learning. The problem of domain adaptation is introduced in the third chapter as well as domain adaptation techniques used in this thesis. The fourth chapter describes datasets used for evaluation, implementation details, and experimental setup along with evaluation results which are discussed. Conclusion and suggestions for future work are given in chapter five.

2. Sentiment Analysis

Surveys of Pang and Lee (2008) and Liu and Zhang (2012) give extensive review of sentiment analysis tasks and research directions. Sentiment analysis can be performed on different levels of granularity. Aspect-based sentiment analysis deals with extracting sentiment expressed towards specific entities and their aspects. This includes the tasks of both identifying entities and their aspects, as well as sentiment expressed towards each aspect. Sentence level sentiment analysis deals with sentiment at the sentence level, where the goal is to determine whether individual sentences are positive or negative. This task is related to sentence subjectivity classification, where the goal is to determine whether sentence is subjective or objective. Coarsest form of sentiment analysis is document level sentiment analysis where, given document, the goal is to determine the overall sentiment of the opinion holder about the entity that is the topic of the document (Liu and Zhang, 2012). Document level sentiment analysis is perhaps the most studied form of sentiment analysis (Liu and Zhang, 2012). Research on document level sentiment analysis implicitly assumes that the given document expresses opinion on single entity and contains opinion from a single opinion holder. Products and services reviews satisfy this assumption, since single review focuses on a single product and is written by a single person. On the other hand, blog and forum posts often discuss and compare multiple entities, which makes it hard to determine overall document sentiment (Liu and Zhang, 2012).

Methods for sentiment analysis can be categorized as unsupervised and supervised (Liu and Zhang, 2012). Supervised methods require a labeled training set, whereas unsupervised ones use only unlabeled data. Most popular unsupervised methods are lexicon-based and involve calculating sentiment polarity for a document from the sentiment orientation of words or phrases in the document (Taboada et al., 2011). Sentiment orientation of words is acquired from sentiment lexicons that usually provide context-independent sentiment information such as positivity, negativity, or objectivity of given word. Supervised methods formulate sentiment analysis as typical text classification task and employ supervised machine learning algorithms in order to learn a classifier which can predict sentiment. Traditional text classification classifies documents of different topics, e.g., politics and sports, where topic-related words are the key features. In sentiment classification more important are sentiment

bearing words, i.e., words that indicate positive or negative opinions, such as *great*, *amazing*, *horrible*, *lame* (Liu and Zhang, 2012).

In order to use the most of supervised learning algorithms, text needs to be represented as an d-dimensional feature vector $\mathbf{x}$. A typical way to represent text is to use bag-of-words model, where elements of a feature vector correspond to words from vocabulary that is inferred from the available documents. Feature x_i is 0 if a word w_i doesn't appear in a document. If the word i appears in document, x_i can be 1 to denote the presence of word or, alternatively x_i can be the frequency of the word in document to denote its importance. This is also known as the "bag-of-unigrams" representation, since features are individual words. Unigrams do not preserve information about word ordering, which may be valuable and to somewhat mitigate this problem one can use n-grams instead of unigrams, i.e., instead of using single words as features, m consecutive words are used. Typically, pre-processing techniques are used on raw text to obtain document representation. A document is firstly tokenized and stopwords are typically removed. Stopwords are common words, such as prepositions, articles, and punctuations, which appear often and typically do not bear valuable information. Additionally, stemming or lemmatisation can be applied as normalization technique for reducing multiple morphological variations of words to single form. Stemming reduces inflected word to its base form whereas lemmatisation reduces a word to its dictionary form.

Pang et al. (2002) were the first to take supervised classification approach to sentiment classification, specifically they classified movie reviews as positive or negative. They experimented with naive Bayes, maximum entropy, and support vector machine (SVM) classifiers. Their experiments show that using unigram features with SVM classifier performs best. Most of the subsequent research on supervised sentiment classification follows the same paradigm, but differs in the features or learning algorithms used (Liu and Zhang, 2012). As is typical for supervised classification, feature engineering proved to be important for sentiment classification, so researchers tried various features and feature selection techniques. Instead of using raw term occurrence or frequency for features, weighting schemes can be used. TF-IDF, which stands for term frequency-inverse document frequency, is a commonly used information retrieval weighting scheme, which tries to account for a fact that some words generally appear often across corpus and are likely to be uninformative. Part-of-speech (POS) of words can be used as a form of crude word sense disambiguation, i.e.,words with different POS tags are treated as separate features. Since adjectives are intuitively an important indicator of sentiment, Pang et al. (2002) experimented with using only adjectives as features, but results do not follow intuition. Following lexicon-based approaches, occurrence of sentiment bearing words such as *excellent* or *horrible* can be used as additional features, since they are often quite indicative of document sentiment. Negations such as *not* and *never* are often senti-

ment shifters, that is they reverse sentiment polarity of words and phrases and they can be leveraged to engineer new features. For example, occurrence of *don't like* results in the feature *NOT-like*, which indicates a negative sentiment. Syntactic relations between words and phrases obtained from parsing or dependency trees were also used as features.

Since some features are redundant or irrelevant for the task at hand, feature selection methods are often used. Examples of commonly used feature selection methods are document frequency, information gain and mutual information (Pang and Lee (2008); Alpaydin (2014)). Document frequency feature selection simply selects words that appear in at least a given number of documents, or alternatively words that appear at least given number of times in the training set. On the other hand, mutual information and information gain are based on information theory and rely on conditional probabilities of words appearing in given class.

Many different learning algorithms were tested by researchers on the task of sentiment classification. Support Vector Machine typically shows strong performance (Pang et al. (2002); Wang and Manning (2012)). Moraes et al. (2013) perform empirical comparison of artificial neural networks and SVM for the task of document-level sentiment classification. They experiment on movie reviews and show that neural networks can outperform SVM. In experiments with unigram features, SVM achieves 83% accuracy, whereas neural networks achieve 86% accuracy. More recently, deep learning techniques were used for sentiment classification. The basis for most of these approaches are word embeddings, which are dense, real-valued vector representations of words such that two words are close in vector space if they are semantically similar (Turian et al., 2010). Word2Vec embeddings (Mikolov et al., 2013) exhibit semantic compositionality and were successfully used in different deep learning approaches for sentence-level sentiment classification: convolutional neural network (Kim, 2014), recursive neural network (Socher et al., 2013), and long-short term memory network (Tai et al., 2015). Semantic compositionality principle states that the meaning of larger units of text is a function of meanings of its parts. This is important, since it means that we can use word embeddings to represent larger units of text. However, using word embeddings to represent something more than a sentence is not straightforward. Le and Mikolov (2014) propose Paragraph Vector, neural network approach for learning document embeddings which can be used in addition to, or in place of, standard bag-of-words features. Using Paragraph Vector embeddings with SVM classifier showed state-of-art results in document-level sentiment classification, around 93% accuracy on movie reviews dataset (Le and Mikolov, 2014).

The remainder of this chapter deals with supervised classification task in the context of sentiment analysis. Section 2.1 introduces supervised classification framework and two commonly used classification algorithms: support vector machines and neural networks. In section 2.2, problem of cross-domain sentiment analysis is introduced and discussed.

2.1. Supervised Classification

Supervised classification is the problem of learning a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ which assigns example $\mathbf{x}$ its class label y (Alpaydin, 2014). Examples come from d-dimensional input space $\mathcal{X}$ and class labels are discrete values, e.g., in binary classification $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, 1\}$. Examples are assumed to be drawn independently from an identical joint distribution $p(\mathbf{x}, y)$, what is known as the i.d.d. assumption (Alpaydin, 2014). Hypothesis is a function $h(\mathbf{x} \mid \theta)$, parametrized with vector of parameters θ. Hypothesis h comes from a model $\mathcal{H}$, also known as hypothesis space. For example, linear classification models consist of a set of hypotheses defined with $h(\mathbf{x} \mid \mathbf{w}, b) = \mathbf{w}^T\mathbf{x} + b$. In other words, linear classifier finds linear decision boundary which divides the input space into two halves. Training a classifier amounts to searching hypothesis space $\mathcal{H}$ for hypothesis h^* which classifies examples most accurately, i.e., one which minimizes the expected error:

$$h^* = \arg\min_{h \in \mathcal{H}} \sum_{(\mathbf{x},y) \in \mathcal{X} \times \mathcal{Y}} p(\mathbf{x}, y) L(y, h(\mathbf{x})), \tag{2.1}$$

where loss function L is a measure of how wrong is classification of given example.
Since true distribution $p(\mathbf{x}, y)$ is unknown, empirical error on training dataset
$D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ is minimized instead:

$$h^* = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{N} L(y_i, h(\mathbf{x}_i). \tag{2.2}$$

In statistical learning theory, this is also known as empirical risk minimization principle (Vapnik and Vapnik, 1998). To determine performance on unseen data, generalization error is computed on separate test set.

Different supervised classification algorithms differ in terms of hypothesis space, loss function and optimization procedure used to minimize empirical error (Alpaydin, 2014). The remainder of this sections introduces two classification algorithms: support vector machine and neural networks. The overview is based on the machine learning textbook by Alpaydin (2014).

2.1.1. Support Vector Machine

Support Vector Machine (SVM) is a binary classifier which finds the optimal separating hyperplane that maximizes margin, i.e., the distance to closest examples on both sides of the hyperplane (Cortes and Vapnik, 1995). In other words, a linear decision boundary is found so that it is as far away as possible from the examples of both classes. Figure 2.1 depicts the optimal separating hyperplane in a 2-dimensional input space. With hyperplane being defined with $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = 0$, if $h(\mathbf{x}) > 0$ example $\mathbf{x}$ is classified as positive and if $h(\mathbf{x}) < 0$ as

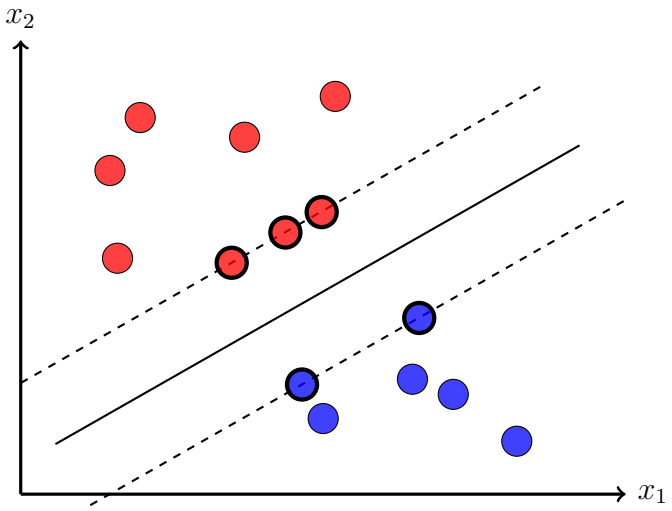

Figure 2.1: Hard-margin SVM optimal separating hyperplane.

negative. Given training set $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, where $y_i \in \{-1, 1\}$, decision boundary should classify all examples correctly with margin being as large as possible. Such decision boundary can be found by solving the following constrained optimization problem:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i. \end{aligned} \tag{2.3}$$

This is the hard-margin formulation of SVM and has a solution only if training set is linearly separable. Soft-margin formulation of SVM also has a solution if the training set is not linearly separable, as is often case with real-world problems. Along with finding the maximum

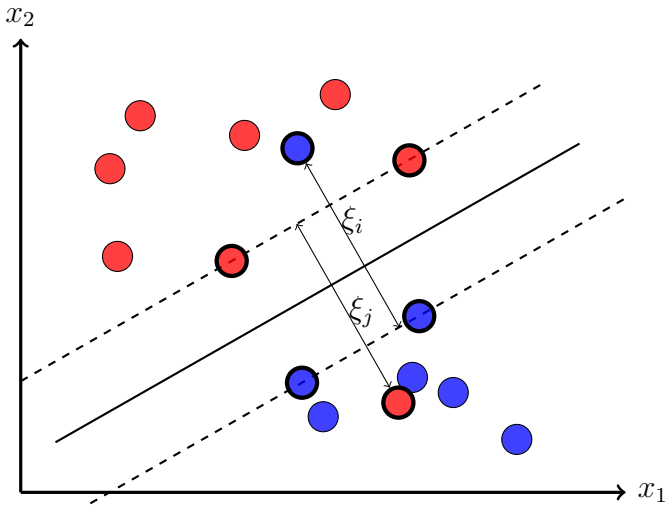

Figure 2.2: Soft-margin SVM optimal separating hyperplane.

margin hyperplane, the goal is to also minimize misclassifications of training examples. To this end, slack variables $\sum_i \xi_i \geq 0$ are introduced as a measure of misclassification, i.e., how

far each example is from correct side of margin, as shown in Figure 2.2. Soft-margin SVM is optimized by solving the following constrained problem:

$$\begin{aligned} \text{min} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i \quad \forall i, \\ & \xi_i \geq 0 \quad \forall i. \end{aligned} \tag{2.4}$$

Hyperparameter C is a regularization factor and dictates the trade-off between margin size and penalty for misclassifications of training examples. Optimization problems in 2.3 and 2.4 can be formulated as instances of quadratic programming and solved using standard methods for quadratic programming. Weights of optimal separating hyperplane can be represented as linear combination of training examples, so SVM model can be written as:

$$h(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \mathbf{x} + b, \tag{2.5}$$

where α_i is non-zero only for support vectors, i.e., training examples which are closest to optimal hyperplane. Support vectors are examples which would be hard to classify were they not in training set.

SVM is a linear model and might be too simple if classification problem is not linearly separable. In order to learn nonlinear decision boundary using linear classifier, a typical approach is to use nonlinear transformation $\Phi : \mathcal{X} \to \mathcal{F}$, which maps the examples into higher dimensional, possibly infinitely, new feature space $\mathcal{F}$, in which, hopefully, the problem will become linearly separable (Alpaydin, 2014). Linear decision boundary in the new space corresponds to the nonlinear decision boundary in original input space. SVM goes ones step further and uses kernel functions instead explicitly mapping examples, a method known as *kernel trick*. A kernel function $k(\mathbf{x}_1, \mathbf{x}_2)$ computes the inner product, which is generalization of a dot product, of two vectors (from input space) in new feature space without explicitly applying the feature transformation. SVM model in equation 2.5 uses a linear kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$, which computes dot product in original input space. A popular choice of kernel function is the radial basis function kernel (Gaussian kernel) defined as $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)$. The RBF kernel computes the similarity between examples based on euclidean distance in input space. For similar examples, the RBF kernel approaches 1 when examples are close in input space and 0 when examples are distant in input space. Hyperparameter γ dictates the spread of Gaussian function or, in other words, how close examples should be to be considered similar. Hyperparameters γ and C control the complexity of the SVM model and should be tuned with care, which is typically done via cross-validation.

2.1.2. Artificial Neural Networks

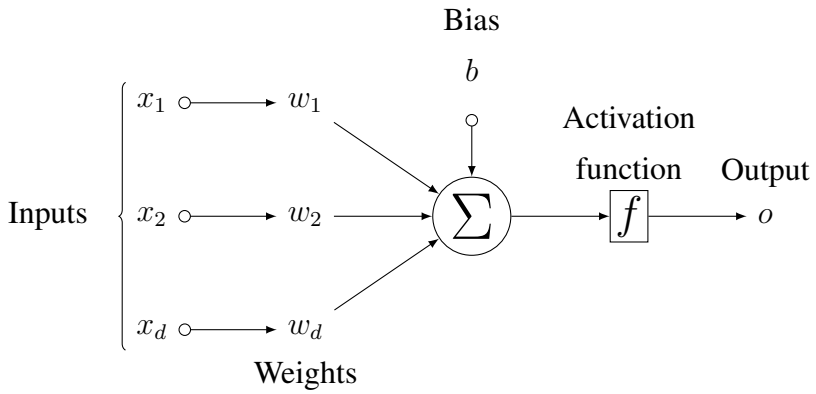

Figure 2.3: Diagram of artificial neuron.

The basic computational unit of neural networks is an artificial neuron, which is shown on Figure 2.3. Artificial neuron is somewhat inspired by biological neuron that produces an output signal based on inputs it receives from other neurons, with each input importance being determined by the strength of connection between corresponding neurons. Similarly, an artificial neuron computes its output o from d inputs. Each input x_i has weight w_i associated with it and weights control influence of inputs. An artificial neuron computes weighted sum of its inputs and passes it through activation function which produces an output signal. An artificial neuron can be modeled with function $o : \mathbb{R}^d \to \mathbb{R}$, defined as:

$$o(\mathbf{x}) = f(\sum_{i=1}^{n} x_i w_i + b) = f(\mathbf{w}^T \mathbf{x}), \tag{2.6}$$

where weight vector $\mathbf{w}$ is a $(d+1)$-dimensional vector with $w_0 = b$, input vector $\mathbf{x}$ is also $(d+1)$-dimensional with $x_0 = 1$, and f is called activation or threshold function. Weights of the artificial neuron are learned in an iterative manner using gradient descent or one of its variants. Gradient descent optimizes objective function $E(\mathbf{w})$ parametrized with weight vector $\mathbf{w}$. In order to use gradient descent for learning, both objective function and activation function need to be differentiable. Weights are initialized at random and iteratively updated in the direction opposite of the gradient of objective function with respect to weights as in equation 2.7, with learning rate η determining the size of update taken. If the learning rate is too low, convergence will be slow, whereas if the learning rate is too high, optimization procedure might diverge and never finish.

$$\mathbf{w}^{new} = \mathbf{w} - \eta \cdot \nabla_{\mathbf{w}} E(\mathbf{w}) \tag{2.7}$$

Gradient of the objective function is estimated from training set either using complete training set (gradient descent), single example (stochastic gradient descent), or part of training

set (mini-batch gradient descent). Typical choice of objective function for classification is cross-entropy loss which, for stochastic gradient descent, can be expressed as:

$$E(\mathbf{w}) = L(\mathbf{x}, y, \mathbf{w}) = -y \log f(\mathbf{w}^T \mathbf{x}_i) - (1 - y) \log f(\mathbf{w}^T \mathbf{x}), \tag{2.8}$$

where $\mathbf{x}$ is an example from training set and y its class label. Weights are iteratively updated until stopping criterion, which is typically defined in terms of convergence of objective function, or alternatively in terms of allowed update steps.

An artificial neuron is a linear model which means it defines a hyperplane that splits the input space into two halves. In order to implement a nonlinear decision boundary, artificial neurons are arranged into layers to form an artificial neural network. Figure 2.4 depicts a feed-forward neural network with one hidden layer. Input layer neurons simply pass inputs

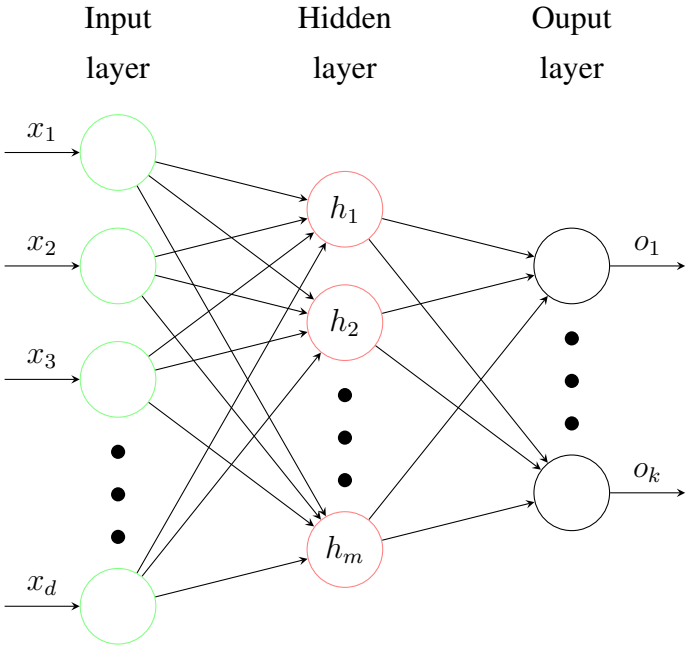

Figure 2.4: Diagram of feed-forward neural network with one hidden layer.

x_i to hidden layer neurons without performing any computation. Each neuron in the hidden layer is associated with a set of weights $\mathbf{w}_i$ and computes its output h_i as:

$$h_i(\mathbf{x}) = f(\mathbf{w}_i^T \mathbf{x}). \tag{2.9}$$

Neurons in the output layer have associated weights $\mathbf{v}_i$ and compute their outputs using outputs of hidden layer neurons:

$$o_i(\mathbf{x}) = f(\sum_{j=1}^{m} v_{ij} h_j(\mathbf{x})). \tag{2.10}$$

If activation functions of hidden neurons are nonlinear, the hidden layer performs a nonlinear transformation of input from d-dimensional input space to m-dimensional space spanned by

hidden neurons. The output layer implements a linear decision boundary in this new space. The linear decision boundary in new space corresponds to a nonlinear decision boundary in input space. Typical choices for nonlinear activation functions are logistic sigmoid and hyperbolic tangent. Weights of the neurons in output layer are also learned in an iterative manner via gradient descent, but with additional use of backpropagation (Rumelhard et al., 1986), which is based on using chain rule for computing gradient of objective function with respect to hidden layer weights.

2.2. Cross-Domain Sentiment Analysis

Opinionated documents come from many different domains and genres. Documents from different domains use different vocabularies, language constructs, and can also have different lengths, which leads to a different distribution of features between domains. Intuitively, one might assume that this will not pose a problem for sentiment classification, since the main indicator of sentiment of a document are often context and domain independent sentiment bearing words, such as *good*, *great*, *worst*, or *horrible*. As it turns out, sentiment classifier trained using documents from one domain, called source domain, often performs poorly on documents from a different, target, domain (Liu and Zhang, 2012).

Brooke (2009) did an experiment where a sentiment classifier, specifically SVM, was trained using unigrams as features. The most positive and the most negative unigrams were extracted and, as one might have expected, universal sentiment bearing words such as *worst*, *waste*, and *unfortunately* were highly negative and *wonderful*, *enjoy*, and *memorable* highly positive. However, some unintuitive and domain-specific words turned out to be important for classification. The mention of *plot*, *director*, or *script* turned out to be an indicator of negative sentiment, whereas *performance*, *ending*, and *flaws* indicated positive sentiment. When such a classifier is used on, e.g., car reviews, its performance will drop since it relies on source domain specific features which most likely will never occur in car domain. Another problem is that some words might bear the opposite sentiment depending on the domain. For example, in electronics domain, *straightforward* expresses positive sentiment as in *straightforward to use*, whereas in books domain it will likely be negative as in *straightforward plot*.

One approach to tackle the problem of cross-domain sentiment classification is to label enough examples from the target domain and use them when training a classifier, but labeling a training set for new target domain requires time and resources. Another way is to employ domain adaptation techniques to more efficiently transfer the knowledge from source to target domain.

3. Domain Adaptation

Domain adaptation considers the supervised classification setting in which training and testing data, i.e., data that the classifier will be used on, are sampled from different distributions (Glorot et al., 2011). In this context, domain is a probability distribution defined over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is input space and $\mathcal{Y}$ is a discrete label space. Let $p(x, y)$ denote the probability distribution and $P(x, y)$ denote probability of observing example $\mathbf{x} \in \mathcal{X}$ with the corresponding label $y \in \mathcal{Y}$. Source domain examples are sampled from distribution $p_s(x, y)$, whereas target domain examples are sampled from distribution $p_t(x, y)$. The typical setting is that one has a labeled set from the source domain $D_s = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ and an unlabeled set from the target domain $D_t = \{\mathbf{x}_i\}_{i=1}^{N'}$. Some work also considers supervised setting in which a number of labeled examples from target domain is available as well (Daumé III, 2007). In domain adaptation setting, we want to minimize the expected error of hypothesis h on the target domain. Since there are no labeled examples from the target domain, we cannot the minimize empirical error on the target domain. Simply minimizing empirical error on the source domain, for which labeled examples are available, leads to a violation of i.d.d. assumption, since $p_s(x, y) \neq p_t(x, y)$. This means that we cannot expected good performance when using such hypothesis on examples from the target domain. Certain assumptions about source and target domains can be made. Joint distribution $p(x, y)$ can be factored as

$$p(x, y) = p(y \mid x)p(x). \tag{3.1}$$

This means that domains can differ in conditional distribution of class labels given example $p(y \mid x)$ and in marginal distribution of examples $p(x)$. Domain adaptation methods generally assume that $p_s(y \mid x) = p_t(y \mid x)$, i.e., conditional probabilities of class labels are the same in both domains. This assumption is needed since for arbitrary $p_s(y|x)$ and $p_t(y|x)$ there is no way to learn good hypothesis without labeled data from target domain (Huang et al., 2007). In this case domains differ only in marginal distribution of examples $p_s(x) \neq p_t(x)$, which is known as a covariate shift (Shimodaira, 2000). There are two main approaches to tackling this domain mismatch: instance weighting approach and feature transformation approach. They are reviewed in sections 3.1 and 3.2, respectively, based on surveys from Jiang (2008) and from Pan and Yang (2010).

3.1. Instance Weighting Approach

Instance weighting approach is based on making some examples from the source domain more important for learning a good hypothesis for the target domain. To see why this is the case, consider finding hypothesis h^* by minimizing expected risk:

$$h^* = \arg\min_{h \in \mathcal{H}} \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} p(x, y) L(y, h(\mathbf{x})). \tag{3.2}$$

Since distribution $p(x, y)$ is unknown, we estimate the expected risk from labeled set $\{(x_i, y_i\}_{i=1}^N$, randomly sampled from $p(x, y)$, and minimize the empirical error to find the hypothesis:

$$h^* = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{N} L(y_i, h(\mathbf{x}_i)). \tag{3.3}$$

In domain adaptation setting, we can minimize empirical error only for the source domain, since labeled set is not available for target domain. However, we can rewrite minimization of the expected risk for the target domain and minimize the weighted empirical risk in source domain as follows (Jiang, 2008):

$$h^* = \arg\min_{h \in \mathcal{H}} \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} p_t(x, y) L(y, h(x)) \tag{3.4}$$

$$= \arg\min_{h \in \mathcal{H}} \sum_{(\mathbf{x},y) \in \mathcal{X} \times \mathcal{Y}} \frac{p_t(x, y)}{p_s(x, y)} p_s(x, y) L(y, h(x)) \tag{3.5}$$

$$\approx \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{N} \frac{P_t(\mathbf{x}_i, y_i)}{P_s(\mathbf{x}_i, y_i)} L(y_i, h(\mathbf{x}_i)). \tag{3.6}$$

The loss for every source domain example $(\mathbf{x}_i, y_i)$ is weighted by ratio $\beta(\mathbf{x}_i, y_i) = \frac{P_t(\mathbf{x}_i, y_i)}{P_s(\mathbf{x}_i, y_i)}$. Instead of learning the hypothesis using only labeled source domain examples, the difference between domains is taken into account. Under covariate shift assumption that distributions $p_s(x, y)$ and $p_t(x, y)$ differ only in marginal distributions $p_s(x) \neq p_t(x)$, the problem reduces to estimating ratio $\beta(\mathbf{x}_i, y_i) = \frac{P_t(\mathbf{x}_i)}{P_s(\mathbf{x})_i}$ for all examples in the source domain labeled set. The loss for examples that are likely in the source domain, but are unlikely in the target domain will have small weights, whereas examples unlikely in source domain, but likely in target domain will have large weights.

Instance weighting approaches differ in how they estimate the ratio $\beta_i = \frac{P_t(\mathbf{x}_i, y_i)}{P_s(\mathbf{x}_i, y_i)}$. One line of work is based on estimating distributions $p_s(x)$ and $p_t(x)$ from available data and then computing β_i for all source domain training examples. This approach has the drawback that estimates for $p_s(x)$ and $p_t(x)$ need to be good and this is hard to achieve with high-dimensional data such as text. Small errors in distribution estimates can lead to large deviations of β. An alternative line of work uses available data to directly estimate β_i. One

such technique is described in the remainder of this section since it is used in experimental part of this thesis.

Kernel mean matching (KMM) is a method for estimation of weights β_i directly by matching source and target distributions. The idea behind KMM is to minimize the mean distance between weighted source domain distribution $\beta(x)p_s(x)$ and the corresponding target domain distribution $p_t(x)$ in reproducing kernel Hilbert space (RKHS) $\mathcal{F}$ with mapping $\Phi : \mathcal{X} \to \mathcal{F}$. The mean distance is measured via maximum mean discrepancy (MMD), which is defined as:

$$\left\| \mathop{\mathbb{E}}_{\mathbf{x}\sim p_s(x)} \Big[\beta(\mathbf{x})\Phi(\mathbf{x})\Big] - \mathop{\mathbb{E}}_{\mathbf{x}\sim p_t(x)} \Big[\Phi(\mathbf{x})\Big] \right\|^2 . \tag{3.7}$$

Under the assumption that $p_t(x)$ is absolutely continuous with respect to p_s, i.e., $P_s(\mathbf{x}) = 0$ implies $P_t(\mathbf{x}) = 0$, and that $\mathcal{F}$ is RKHS induced by the universal kernel $k(\mathbf{x_i}, \mathbf{x_j}) = \Phi(\mathbf{x_i})\Phi(\mathbf{x_j})$, minimizing MMD converges to $p_t(x) = \beta(x)p_s(x)$ (Gretton et al., 2009). Weights β_i can be approximated empirically, given N_s source domain examples and N_t target domain examples, by minimizing empirical MMD:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \left\| \frac{1}{N_s}\sum_{i=1}^{N_s} \beta_i \Phi(x_{s_i}) - \frac{1}{N_t}\sum_{i=1}^{N_t} \Phi(x_{t_i}) \right\|^2 . \tag{3.8}$$

Minimizing empirical MMD in equation 3.8 with respect to β is an instance of quadratic programming problem. Let $\mathbf{K}$ be $N_s \times N_s$ kernel matrix computed from source domain examples with $\mathbf{K}_{ij} = k(\mathbf{x}_{s_i}, \mathbf{x}_{s_j})$, $\mathbf{k}$ be N_s dimensional vector with $k_i = \frac{N_s}{N_t}\sum_{i=1}^{N_t} k(\mathbf{x}_{s_i}, \mathbf{x}_{t_j})$, and $\boldsymbol{\beta}$ be N_s dimensional vector. Quadratic programming problem can be formulated as:

$$\begin{aligned} \underset{\beta}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{\beta}^T\mathbf{K}\boldsymbol{\beta} - \mathbf{k}^T\boldsymbol{\beta} \\ \text{subject to} \quad & \beta_i \in [0, B] \\ & \left| \sum_{i=1}^{N_s} \beta_i - N_s \right| \leq N_s\epsilon. \end{aligned} \tag{3.9}$$

The first constraint provides robustness by limiting weights, i.e., the influence of individual examples, whereas the second constraint ensures that $\beta(x)p_s(x)$ is close to probability distribution $p_t(x)$. The authors propose to select ϵ as $\frac{B}{\sqrt{N_s}}$. Quadratic problem in equation 3.9 is convex and can be solved efficiently with quadratic programming solvers to find instance weights β_i for examples from the source domain training set.

3.2. Change-of-Representation Approach

The second approach to domain adaptation is based on changing representation of examples, i.e., transforming examples from input space $\mathcal{X}$ to feature space $\mathcal{F}$. Changing the representation can potentially affect both the marginal distribution $p(x)$ as well as conditional

distribution $p(y \mid x)$. If we can find a feature transformation $\Phi : \mathcal{X} \rightarrow \mathcal{F}$ under which $p_s(x, y) = p_t(x, y)$, we no longer have a domain adaptation problem since source and target domain examples are sampled from same joint distribution. Ben-David and Blitzer (2007) dealt with generalization bounds for domain adaptation and gave one of the first theoretical treatments of effect of feature representations in domain adaptation setting. Specifically, they gave bound for empirical error of hypothesis learned in the source domain and evaluated in the target domain. They show that generalization error in the target domain is bound by two terms. The first term bounds the error with performance of the hypothesis in the source domain, which means that there should exists a good hypothesis for the source domain in order to expect to learn a hypothesis that performs well on target domain. The second term bounds the error with the divergence of domains, i.e., divergence between marginal distributions $p_s(x)$ and $p_t(x)$ in feature space $\mathcal{F}$. The authors propose $\mathcal{H}$-divergence, which relies on the capacity of hypothesis class to distinguish whether the given example comes from source or from target domain, as a measure of divergence between source and target marginal distributions. If domain adaptation by changing of representation is to be successful, new representation of examples should be domain-invariant while remaining discriminative in source domain. The benefit of the representation approaches to domain adaptation is that they can be used with any standard learning algorithm. Examples from both domains are simply transformed to a new feature space in which any classifier can be trained on labeled data from the source domain. The remainder of this section describes domain adaptation based on change of representations used in experimental part of this thesis.

3.2.1. Frustratingly Easy Domain Adaptation

Frustratingly easy domain adaptation (FEDA) (Daumé III, 2007) is a supervised approach to domain adaptation, i.e., the availability of N_t labeled examples from target domain is assumed along with N_s labeled examples from the source domain with typically $N_s >> N_t$. Since we have some labeled data in the target domain, we might try to use union of labeled source and target domain examples to learn a classifier that will perform better on the target domain than a classifier learned only on the source domain. The authors experimentally show that applying FEDA can beat this simple baseline. FEDA is an extremely simple domain adaptation method based on changing feature representation. FEDA augments original input space with new features. Essentially, each feature from input space is taken and three new versions are made: a general version, a source specific version, and a target specific version. Examples from source domain will contain only general and source specific features, whereas target domain examples will contain only general and target specific features. Let input space be $\mathcal{X} = \mathbb{R}^n$ and augmented feature space be $\mathcal{X}^* = \mathbb{R}^{3n}$. We can then define mappings $\Phi_s, \Phi_t : \mathcal{X} \rightarrow \mathcal{X}^*$ for source domain and target domain examples, respectively.

Mappings are simply concatenations of the input space feature vectors and n-dimensional zero feature vector $\mathbf{0}$:

$$\Phi_s(\mathbf{x}) = [\mathbf{x}, \mathbf{x}, \mathbf{0}], \tag{3.10}$$

$$\Phi_t(\mathbf{x}) = [\mathbf{x}, \mathbf{0}, \mathbf{x}]. \tag{3.11}$$

Before learning a classifier, examples from the source domain are transformed using Φ_s and examples from target domain are transformed using Φ_t. Authors give an intuitive insight into why applying these mappings helps reducing the error on the target domain. It can be thought as simultaneously learning two classifiers, one for combination of source specific and general features and another for combination of target specific and general features. Having separate versions of features enables the classifier to give different importance to features depending on which domain examples is coming form. For example, in case of sentiment classification, consider negative examples that contain *straightforward plot* from movie domain and positive examples that contain *straightforward to use* for electronics domain. The general version of unigram *straightforward* might have a low weight since it bears the opposite sentiment in source and target domains. Movie and electronics version of *straightforward* might have high negative and positive weights, respectively, since in the movie domain it is correlated with negative sentiment, and in electronics with positive sentiment.

3.2.2. Structural Correspondence Learning

Structural correspondence learning (SCL) is a domain adaptation approach which creates new representation which tries to bridge the gap between source and target domains by learning feature correspondence between the two domains (Blitzer et al., 2007). Consider smartphone and restaurant reviews. Both domains will contain universally positive and negative words like *good* or *bad*. Other words are domain specific, e.g., *speaker* and *waiter*. The intuition is that, although "loud speaker" and "fast waiter" occur only in specific domains, if they are highly correlated with the word *good*, we can assume that they have similar sentiment, i.e., they correspond.

The first step in SCL is to find a set of m pivot features. The pivot features are selected heuristically. Pivots should be frequent in both domains and should behave the same in both domains. In sentiment classification, good pivots would be universally positive or negative words, i.e., words whose sentiment remains constant accross domains. The authors propose selecting pivot features based on high mutual information with source domain labels, i.e., features which are highly informative of sentiment in source domain are selected as pivots. SCL models correlation between pivot features and non-pivot features. New classification problems are created using unlabeled sets from both domains. For each pivot feature, the feature is masked from the unlabeled set and a linear classifier is is trained to predict the

occurrence of pivot in the given example. Elements of weight vector $\mathbf{w}_i$ of i-th pivot predictor correspond to correlation between i-th pivot feature and the rest of the features. Positive entries in vector $\mathbf{w}_i$ means that the corresponding non-pivot feature is highly correlated with i-th pivot feature. If two non-pivot features are correlated with many of same pivot features, they are said to correspond. Pivot predictor weight vectors are arranged into columns of $d \times m$ matrix $\mathbf{W}$, where d is the total number of features. We could use matrix $\mathbf{W}$ to define projection $\mathbf{W}^T\mathbf{x}$ to obtain new m new features, however authors for both computational and statistical reasons decide on low-dimensional approximation approximation of the original feature space. To this end singular value decomposition (SVD) is used. Singular value decomposition (SVD) factorizes given matrix A into orthogonal matrices U and V, and diagonal matrix D so that

$$A = UDV^T. \tag{3.12}$$

Columns of U are left singular vectors, columns of V are right singular vectors of A, and diagonal elements of D are corresponding singular values. Matrix $\mathbf{W}$ is factorized using SVD and projection into k dimensional space is obtained with $h \times d$ matrix θ whose rows correspond to top left singular vectors from U. Mapping Φ simply concatenates original features with features obtained using projection θ:

$$\Phi(\mathbf{x}) = [\mathbf{x}, \theta\mathbf{x}]. \tag{3.13}$$

In order to train a classifier for the target domain, the source domain labeled set is transformed into augmented feature space Φ before training. If θ contains meaningful correspondences, then the classifier trained in augmented feature space should perform well in both source and target domains (Blitzer et al., 2007).

3.2.3. Marginalized Stacked Denoising Autoencoders

Marginalized stacked denoising autoencoders are, essentially, computationally effective approximation of the Stacked Denoising Autoencoder (SDA). SDAs were introduced by Vincent et al. (2010) and were shown to be useful for learning robust representations of data. As the name implies, SDA consists of multiple denoising autoencoders (DA), which are chained so that each DA uses output of DA from previous layer. Denoising autoencoder is an extension of an autoencoder, which is a neural network which learns to encode its input to hidden layer representation that captures meaningful patterns in data. An autoencoder consists of an input layer $\mathbf{x}$, hidden layer $\mathbf{y}$, and output layer $\mathbf{z}$ and can be defined as (Vincent et al., 2010):

$$\mathbf{y} = f_\theta(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{3.14}$$

$$\mathbf{z} = g_{\theta'}(\mathbf{y}) = s(\mathbf{W}'\mathbf{y} + \mathbf{b}'), \tag{3.15}$$

where $f_\theta(\mathbf{x})$ is encoder parameterized by θ: $d' \times d$ weight matrix $\mathbf{W}$ and d' dimensional bias term $\mathbf{b}$, $g_{\theta'}(\mathbf{y})$ is decoder parameterized by θ': $d \times d'$ weight matrix $\mathbf{W}'$ and d dimensional bias term $\mathbf{b}'$, and s is non-linear activation function such as logistic sigmoid. The encoder maps an input $\mathbf{x} \in \mathbb{R}^d$ to a lower-dimensional hidden representation $\mathbf{y} \in \mathbb{R}^{d'}$ and the decoder tries to reconstruct the input from this hidden representation. The hidden representation has a lower dimensionality than the input space so that learning of an identity function is avoided. Training an autoencoder amounts to finding parameters θ and θ' so that reconstruction error $L_H(\mathbf{x}, \mathbf{z})$ is minimized. Since the dimensionality of the hidden layer is lower than the dimensionality of the input layer, the autoencoder is forced to learn hidden representation which is representative of the input. Denoising autoencoder (DA) is an extension of autoencoder which tries to reconstruct input $\mathbf{x}$ from its corrupted version $\hat{\mathbf{x}}$ which is obtained by setting elements of input vector $\mathbf{x}$ to 0 with probability p. The autoencoder and denoising autoencoder are trained via backprogpagation and stochastic gradient descent. Stacked Denoising Autoencoder (Vincent et al., 2010) is obtained by stacking DAs so that the j-th autoencoder is trained based on hidden representation of the $(j-1)$-th layer. Training the SDA is done in greedy manner. The first layer is trained to reconstruct input from its corrupted version. When the first layer is trained, the second layer is trained to reconstruct corrupted hidden representation of first layer, and so on for subsequent layers. This way we get several levels of representations of the input vector. Glorot et al. (2011) applied SDA on domain adaptation problem for sentiment analysis. They trained SDAs on unlabeled data from both source and target domain and used concatenation of hidden representations as features to train sentiment classifier on a source domain that was tested on a target domain. The SDA is able to distangle hidden factors which explain variation in input sample and automatically group features based on those factors (Chen et al., 2012). This helps domain adaptation since those hidden factors are domain invariant. However, the SDA has a lot parameters and learning them via backpropagation can take a long time.

Chen et al. (2012) propose marginalized Stacked Denoising Autoencoder (mSDA) in order to approximate representations learned by the SDA with lower training cost. The idea is to simulate infinitely many different corruptions of inputs, which is refered to as marginalization of explicit corruption. The encoder and decoder are replaced by a single mapping $\mathbf{W}$ which should minimize the squared reconstruction loss:

$$\frac{1}{2n}\sum_{i=1}^{n}||\mathbf{x}_i - \mathbf{W}\hat{\mathbf{x}}_i||^2, \tag{3.16}$$

where $\hat{\mathbf{x}}$ is a stochastically corrupted version of input $\mathbf{x}$ where each feature is corrupted with probability p. A constant feature, which is not corrupted, is added to $\mathbf{x}$; $\mathbf{x} = [\mathbf{x}, 1]$ and also an appropriate bias term is incorporated within the mapping, $\mathbf{W} = [\mathbf{W}, \mathbf{b}]$. Minimizing loss defined in equation 3.16 depends on which features of each input were corrupted. So, instead

of minimizing loss with single corrupted version for each input, multiple passes over training set are done, each time with different corruptions. So the goal becomes to find a mapping $\mathbf{W}$ which minimizes the overall squared loss:

$$\mathcal{L}_{sq}(\mathbf{W}) = \frac{1}{2mn}\sum_{i=1}^{m}\sum_{i=1}^{n}||\mathbf{x}_i - \mathbf{W}\hat{\mathbf{x}}_{ij}||^2, \tag{3.17}$$

where $\hat{\mathbf{x}}_{ij}$ is j-th corrupted version of original input $\mathbf{x}$. By introducing a design matrix $\mathbf{X} = [\mathbf{x_1}, ..., \mathbf{x_n}] \in \mathbb{R}^{d\times n}$, its m-times repeated version $\overline{\mathbf{X}} = [\mathbf{X}, ..., \mathbf{X}]$ and corrupted version of $\overline{\mathbf{X}}$ as $\hat{\mathbf{X}}$, overall squared loss can be rewritten as:

$$\mathcal{L}_{sq}(\mathbf{W}) = \frac{1}{2mn}tr\Big[(\overline{\mathbf{X}} - \mathbf{W}\hat{\mathbf{X}})^T(\overline{\mathbf{X}} - \mathbf{W}\hat{\mathbf{X}})\Big]. \tag{3.18}$$

Minimizing the objective function (3.18) can be expressed as the closed-form solution for ordinary least squares $\mathbf{W} = \mathbf{P}\mathbf{Q}^{-1}$ where $\mathbf{Q} = \hat{\mathbf{X}}\hat{\mathbf{X}}^T$ and $\mathbf{P} = \overline{\mathbf{X}}\hat{\mathbf{X}}^T$.

We are interested in the limit case when $m \to \infty$, i.e., we want to use infinitely many corrupted versions of inputs. In this case, $\mathbf{Q}$ and $\mathbf{P}$ converge to their expected values and mapping $\mathbf{W}$ can be expressed as:

$$\mathbf{W} = E[\mathbf{P}]E[\mathbf{Q}]^{-1}. \tag{3.19}$$

The expectation of $\mathbf{Q}$ can be written as:

$$E\left[\mathbf{Q}\right] = \sum_{i=1}^{n} E\left[\hat{\mathbf{x}}_i\hat{\mathbf{x}}_i^T\right]. \tag{3.20}$$

Off-diagonal entry $\alpha\beta$ of the matrix $\hat{\mathbf{x}}_i\hat{\mathbf{x}}_i^T$ is uncorrupted if both features α and β remained uncorrupted, which happens with a probability of $(1-p)^2$. Diagonal entries remain uncorrupted with probability $(1-p)$, since they depend on corruption of single feature, which happens with probability p. Similarly, expectation of $\mathbf{P}$ can be written as:

$$E[\mathbf{P}] = \sum_{i=1}^{n} E[\mathbf{x}_i\hat{\mathbf{x}}_i^T] \tag{3.21}$$

In this case both off-diagonal and diagonal entries $\alpha\beta$ of matrix $\mathbf{x}_i\hat{\mathbf{x}}_i^T$ remain uncorrupted with probabilty $(1-p)$, since only the second term is corrupted. Let vector $\mathbf{q} = [1-p, ..., 1-p, 1]^T \in \mathbb{R}^{d+1}$ where $\mathbf{q}_\alpha$ represents probabilty of a feature α remaining uncorrupted with last element corresponding to constant feature which is never corrupted. Additionally, scatter matrix of the original features is expressed as $\mathbf{S} = \mathbf{X}\mathbf{X}^T$. Now expectations of matrices $\mathbf{Q}$ and $\mathbf{P}$ can be defined as:

$$E[\mathbf{Q}]_{\alpha\beta} = \begin{cases} \mathbf{S}_{\alpha\beta}\mathbf{q}_\alpha\mathbf{q}_\beta & \text{if } \alpha \neq \beta \\ \mathbf{S}_{\alpha\beta}\mathbf{q}_\alpha & \text{if } \alpha = \beta \end{cases} \tag{3.22}$$

and

$$E[\mathbf{P}]_{\alpha\beta} = \mathbf{S}_{\alpha\beta}\mathbf{q}_{\beta} \tag{3.23}$$

Using closed-form solutions for expectations of matrices $\mathbf{Q}$ and $\mathbf{P}$ we can compute the mapping $\mathbf{W}$ directly without explicitly corrupting inputs. This is referred to as marginalized denoising autoencoder (mDA). Unlike DA, where hidden layer provides hidden representation of an input, mDA doesn't learn to encode data, since goal of mapping $\mathbf{W}$ is to reconstruct corrupted input. However, applying the mapping $\mathbf{W}$ to uncorrupted input might enrich it with features which appear to be missing, leading to better feature representations.

Two factors to which SDA owes it success are nonlinearity, which is incorporated in the encoder function, and stacking of multiple layers of DAs to learn multiple levels of feature representations. Nonlinearity is injected into mDA after computing the mapping $\mathbf{W}$, which preserves closed-form solution. Specifically, a nonlinear squashing function is applied on the output of each mDA. The authors propose using hyperbolic tangent. mDAs can also be stacked to form mSDA by feeding the output of $(t-1)$-th mDA (with applied nonlinearity) as the input of t-th mDA. With the output of t-th mDA denoted by $\mathbf{h}^t$ and original input as $\mathbf{h}^0 = \mathbf{x}$, each mapping $\mathbf{W}^t$ is learned so it reconstructs the output of mDA from the previous layer $\mathbf{h}^{t-1}$:

$$\mathbf{h}^t = s(\mathbf{W}^t\mathbf{h}^{t-1}) \tag{3.24}$$

In order to use mSDA for domain adaptation, unlabeled data from both source and target domains are used for training of an mSDA. Then nonlinearity is applied to outputs of all layers and newly obtained features are used in combination with the original features. More precisely, the original features $\mathbf{x}$ are concatenated with outputs of all layers of mSDA $s(\mathbf{W}^t\mathbf{h}^{t-1})$ to obtain new multiple new representations, one for each mDA. Using such augmented representation, classifier can be trained on a labeled set from the source domain and successfully used to classify examples from the target domain. Chen et al. (2012) report that mSDA matches SDA performance on sentiment analysis domain adaptation while being significantly faster with mSDA taking minutes to train whereas SDA training can take days.

3.2.4. Domain Adversarial Neural Networks

A domain adversarial neural network (DANN) is an approach which embeds domain adaptation into the process of training a feed-forward neural network (Ajakan et al. (2014); Ganin et al. (2016)). The hidden layers of typical neural networks learn hidden representations of input which is discriminative for the task at hand. In addition to learning discriminative representation, DANN also learns a representation that is invariant to the change of domains. A trained DANN can be used directly on a target domain without being affected by the differences in domains. DANN is motivated by theoretical treatment of domain adaptation

by Ben-David and Blitzer (2007), which proves that error of domain adaptation via change of representation is bound with $\mathcal{H}$-divergence, which is a measure of discrepancy between domains. Ben-David and Blitzer (2007) propose a simple approach to approximate of $\mathcal{H}$-divergence; one can simply train and evaluate a classifier for the task of distinguishing if a given example comes from the source or target domain. If such a classifier achieves a low generalization error, it means that it can distinguish between the domains, which means high $\mathcal{H}$-divergence. On the other hand, if the error is high, it means that the classifier is unable to distinguish between the domains, meaning low $\mathcal{H}$-divergence. Good representation for domain adaptation is the one which makes it hard to train a classifier to distinguish between the domains. This is the core idea behind DANN.

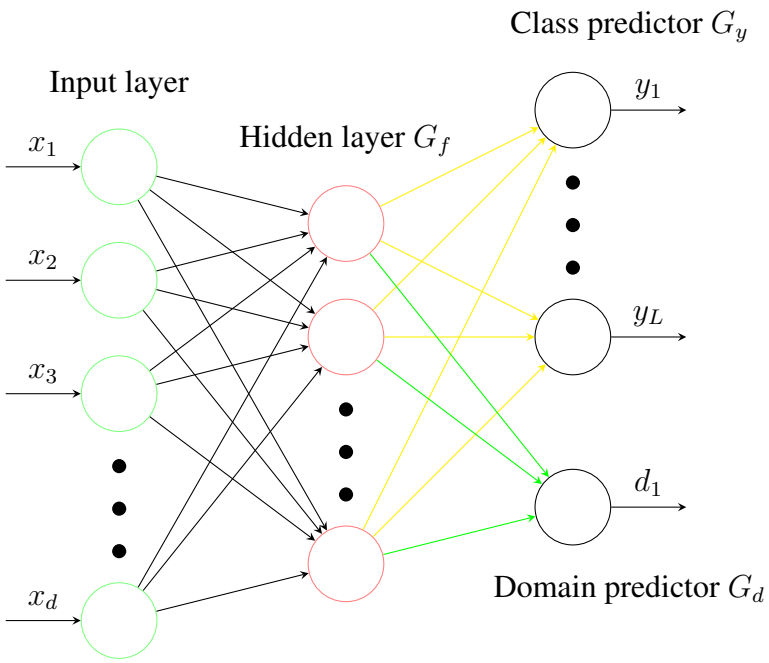

Figure 3.1: A diagram of a shallow domain adversarial neural network with one hidden layer.

Figure 3.1 depicts a shallow domain adversarial neural network. The first part of a DANN is a hidden layer G_f that learns a function $G_f : \mathbb{R}^d \rightarrow \mathbb{R}^D$ which maps examples from a d-dimensional input space into a new D-dimensional representation. The hidden layer G_f is parameterized by weights $\mathbf{W} \in \mathbb{R}^{D \times d}$ and a bias $\mathbf{b} \in \mathbb{R}^D$:

$$G_f(\mathbf{x}|\mathbf{W}, \mathbf{b}) = sigm(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{3.25}$$

where $sigmoid(\mathbf{a})$ applies a sigmoid function to elements of $\mathbf{a}$. The class predictor G_y learns a function $G_y : \mathbb{R}^D \rightarrow [0, 1]^L$ which is parameterized by weights $\mathbf{V} \in \mathbb{R}^{L \times D}$ and bias $\mathbf{c} \in \mathbb{R}^L$:

$$G_y(G_f(\mathbf{x})|\mathbf{V}, \mathbf{c}) = softmax(\mathbf{V}G_f(\mathbf{x}) + \mathbf{c}) \tag{3.26}$$

with $softmax(\mathbf{a}) = \left[\frac{exp(a_i)}{\sum_j exp(a_j)}\right]_{i=1}^{|\mathbf{a}|}$. The softmax function squashes the elements of a vector

to $[0, 1]$, so each component of $G_y(G_f(\mathbf{x}))$ can be viewed as a conditional probability that the neural network assigns $\mathbf{x}$ to each of L classes. Given a source domain example $(\mathbf{x_i}, y_i)$, the negative log-probability of the correct label is used as the classification loss:

$$\mathcal{L}_y(G_y(G_f(\mathbf{x}_i)), y_i) = -\log\Big[G_y(G_f(\mathbf{x}_i))_{y_i}\Big] \tag{3.27}$$

Training of the class predictor can be done by minimizing the following objective:

$$\min_{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}}\left[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}) + \lambda R(\mathbf{W},\mathbf{b})\right], \tag{3.28}$$

with $\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c})$ being a shorthand for prediction loss on i-th example. $R(\mathbf{W},\mathbf{b})$ is an regularizer weighted by hyperparameter λ.

The main idea of DANN is to design a regularizer which will not only help with overfitting, but also with domain adaptation. Domain predictor G_d is used to aid the hidden layer with learning domain invariant features. Domain predictor $G_d : \mathbb{R}^D \to [0, 1]$ is basically a logistic regression classifier parametrized by a vector $\mathbf{u}$ and scalar z which maps hidden representation G_f of input example $\mathbf{x}$ to the probability that a given example is from the target domain:

$$G_d(G_f(\mathbf{x})|\mathbf{u}, z) = sigm(\mathbf{u}^T G_f(\mathbf{x}) + z). \tag{3.29}$$

The loss for domain predictor is the cross-entropy loss defined as:

$$\mathcal{L}_d(G_d(G_f(\mathbf{x}_i)), y_i) = -d_i \log\Big[G_d(G_f(\mathbf{x}_i))\Big] - (1 - d_i)\log\Big[1 - G_d(G_f(\mathbf{x_i}))\Big], \tag{3.30}$$

where d_i is the domain label with $d_i = 0$ if $\mathbf{x_i}$ comes from the source domain, or $d_i = 1$ if $\mathbf{x_i}$ comes from the target domain. The domain predictor is used as a domain regularizer by minimizing:

$$R(\mathbf{W},\mathbf{b}) = \max_{\mathbf{u},z}\Big[-\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z) - \frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z)\Big], \tag{3.31}$$

were average loss is computed on examples from the source domain and unlabeled examples from the target domain. Plugging in the domain regularizer into the objective for training a class predictor from equation (3.28), we obtain:

$$E(\mathbf{W},\mathbf{V},\mathbf{b},\mathbf{c},\mathbf{u},z) = \\ \min_{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}}\left[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}) - \lambda\max_{\mathbf{u},z}\Big[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z) + \frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z)\Big]\right]. \tag{3.32}$$

This optimization objective involves minimization with respect to some parameters as well as a maximization with respect to others. In general, the objective is being maximized with

respect to θ_f parameters of the hidden layer, i.e., feature extractor, and θ_y parameters of class predictor, whereas it is being minimized with respect to θ_d parameters of the domain predictor. The objective is maximized by finding a saddle point defined by $\hat{\theta_f}$, $\hat{\theta_y}$, $\hat{\theta_d}$ such that:

$$\begin{aligned}(\hat{\theta_f}, \hat{\theta_y}) &= \arg\min_{\theta_f, \theta_y} \quad E(\theta_f, \theta_y, \hat{\theta_d}) \\ (\hat{\theta_d}) &= \arg\max_{\theta_d} \quad E(\hat{\theta_f}, \hat{\theta_y}, \theta_d).\end{aligned} \tag{3.33}$$

The authors propose finding the stationary point by doing the following stochastic gradient descent updates:

$$\begin{aligned}\theta_f^{new} &= \theta_f - \eta\Big(\frac{\partial L_y}{\partial \theta_f} - \lambda \frac{\partial L_d}{\partial \theta_f}\Big), \\ \theta_y^{new} &= \theta_y - \eta\Big(\frac{\partial L_y}{\partial \theta_y}\Big), \\ \theta_d^{new} &= \theta_d - \eta\Big(\frac{\partial L_d}{\partial \theta_d}\Big),\end{aligned} \tag{3.34}$$

where η is learning rate. The DANN basically consists of two neural networks: a label predictor and a domain predictor, which compete against each other in an adversarial way. Gradient update rule for hidden layer parameters θ_f in equation (3.34) consists of gradients of class and domain predictors, however, instead being summed they are subtracted. The label predictor is trained using labeled examples from the source domain to update hidden layer parameters such that hidden representation is discriminative in the source domain. On the other hand, the domain predictor is trained using unlabeled examples from both domains for predicting the domain of origin for a given example. Instead of updating hidden layer parameters θ_f so that hidden representation is discriminative for prediction of domains, the domain predictor will update parameters θ_f so that the hidden representation becomes indistinguishable between the domains. The authors introduce a special gradient reversal layer (GRL) which enables easy implementation of DANN in most libraries for deep learning, which typically require layers to define procedures for the forward pass and backpropagation. The GRL is a layer with no parameters associated with it. During forward propagation, the GRL simply does identity transformation. During backpropagation, the GRL takes the gradient from subsequent layer and multiplies it by -1, before passing it to preceding layer. To help with domain adaptation, the GRL can be inserted after the feature extraction layer in almost any neural network architecture. The authors also evaluated a shallow DANN on the task of sentiment classification and it helped with domain adaptation in most of the evaluated adaptation tasks.

4. Experiments

The goal of this thesis was to implement a sentiment classification system which should be able to work on the domain for which we don't have unlabeled samples. To this end, a support vector machine was used as classifier since it was shown that it performs well in the task of sentiment classification (Pang et al., 2002). In order to accomplish domain adaptation, several domain adaptation techniques were tried out: frustratingly easy domain adaptation (Daumé III, 2007), structural correspondence learning (Blitzer et al., 2007), marginalized stacked denoising autoencoders (Chen et al., 2012), domain adversarial neural networks (Ajakan et al., 2014), and kernel mean matching (Huang et al., 2007). Evaluations of implemented classification systems were conducted on both Croatian and English datasets. This chapter describes the datasets, implementation, the experimental setup, and discusses the obtained results.

4.1. Datasets

To evaluate domain adaptation techniques, we needed both labeled and unlabeled data from multiple domains. The standard English dataset for domain adaptation consists of Amazon product reviews for different product types: books, DVDs, electronics, and kitchen appliances (Blitzer et al., 2007). In addition to writing reviews, Amazon provides users with ability to rate products on a scale from 1 to 5. These ratings were used to obtained sentiment labels; reviews with rating over 3 were labeled as positive, whereas reviews with rating less 3 as negative. The rest of the reviews were discarded because of their ambiguity. Table 4.1 shows the number of reviews per domain as well as the distribution of labels. All domains are fairly balanced in terms of positive an negative reviews, which is desirable when training classifiers.

Unlike the English dataset, the Croatian dataset consists of two domains and is a bit more heterogenous. One domain consists of restaurant reviews acquired from a food ordering website and used for aspect-oriented sentiment analysis (Glavaš et al., 2013). Each review is accompanied by an opinion rating on a scale from 0.5 to 6 that the authors used to acquire labels; reviews with rating lower than 2.5 were labeled as negative and reviews with rating

	All	Positive	Negative
Kitchen	7945	3945	3991
DVD	5586	2807	2779
Electronics	7681	3857	3824
Books	6465	3264	3201

Table 4.1: Distribution of labels per domain for English dataset

higher than 4 as positive. The rest of the ratings are mostly inconsistent as they are assigned to both positive and negative reviews (Glavaš et al., 2013). The second domain consists of comments acquired from the Facebook page of telecommunications companies. Comments, as well as replies to them, were manually annotated with sentiment score on scale from -2.5 (completely negative) to 2.5 (completely negative). Instances with score less than -0.5 were labeled as negative and instances with score higher than 0.5 as positive. As before, instances with scores between -0.5 and 0.5 were discarded.

	All	Positive	Negative
Restaurants	2158	1658	500
Telecom	7941	1293	6648

Table 4.2: Distribution of labels per domain for Croatian dataset

Table 4.2 gives the number of instances and the distribution of labels for each domain. Notice, unlike the domains from the English dataset, which are balanced, the restaurant domain is overwhelmingly positive whereas telecom domain is prevalently negative.

4.2. Implementation

The implementation of sentiment classification system and domain adaptation techniques was done using Python programming language.[1] It was chosen due to its simplicity and availability of various useful open source libraries for scientific computing. NumPy and SciPy are libraries which provide efficient implementations of multi-dimensional dense and sparse matrices and various mathematical operations to use on these matrices (Walt et al., 2011). Scikit-learn is a library which provides implementation of popular machine learning algorithms as well as various related utility methods such as easy computation of evaluation metrics (Pedregosa et al., 2011).

[1]Python Software Foundation. Python Language Reference, version 2.7. Available at `http://www.python.org`

`LinearSVC` from Scikit-learn is a wrapper for efficient implementation of a linear SVM and was used for training sentiment classifiers. `SGDClassifier` is an implementation of stochastic gradient descent and was used to learn pivot predictors by minimizing the Huber loss in the implementation of the SCL (cf. Section 3.2.2). The SVD decomposition step of SCL was done using sparse implementation of SVD from the SciPy library. The mSDA (cf. Section 3.2.3) was implemented based on available implementation from the authors.[2] CVXOPT[3] is a Pyhton library for solving convex optimization problems which was used for solving quadratic programming problem in KMM. TensorFlow (Abadi et al., 2016) is a Python library that enables users to model arbitrary computation through flow graph and is extensively used for efficient training of neural networks. It is straightforward to implement a gradient reversal layer in TensorFlow as it is simply a node in the graph that does not do any computation in the forward pass, but reverses the direction of the gradient during backpropagation. The DANN (cf. Section 3.2.4) was implemented using TensorFlow and a gradient reversal layer.

4.3. Experimental Setup

We evaluate the performance of domain adapted sentiment classifiers on all pairs of domains in both Croatian and English languages. Given the source domain and target domain, we want to learn a classifier for the target domain using unlabeled sets from both domains, and the labeled set from source domain. Each domain is split into two sets: an unlabeled set and a labeled set. The classifier is trained using a labeled set from the source domain and evaluated against the labeled set from the target domain. In addition to the labeled set from the source domain, domain adaptation can be performed using unlabeled sets from both domains. For each domain of the English dataset, 1000 positive and 1000 negative examples are randomly selected as labeled set and the rest are used for the unlabeled set. For each domain of the Croatian dataset, 1000 examples are selected as labeled set via stratified sampling so that distribution of labels within labeled set is preserved.

Datasets from both languages are tokenized, punctation marks removed, and additionally Croatian language dataset is lemmatized, i.e., words are reduced to their dictionary form. This has been shown to help since as normalization technique since Croatian is a highly inflectional language, which negatively affects classification accuracy, and lemmatization can remedy this issue (Rotim and Šnajder, 2017). Lemmatization was done using the CST lemmatizer for Croatian (Željko Agić et al., 2013).

Labels from the unlabeled set are used to to establish a gold standard, i.e., the upper per-

[2] `https://www.cs.cornell.edu/~kilian/code/code.html`
[3] `http://cvxopt.org/`

formance limit for a given domain. A linear SVM is trained using the unlabeled set from the given domain and evaluated on the labeled set of the same domain. This is referred to as "in-domain evaluation" (Blitzer et al., 2007). A binary bag-of-unigrams feature representation was used for all domains and 5000 most frequent unigrams from unlabeled set were used as features. Preliminary experiments showed that using more features did not considerably help with performance, so the number was kept at 5000 for all domains.

The protocol for domain adaptation scenario is as follows. Based on preliminary experiments with in-domain classifiers, 5000 most frequent unigrams are selected from the corresponding unlabeled sets of both domains and their union is used as feature set for binary bag-of-words feature representation of documents in all experiments. The resulting number of features for all pairs of domains in the English dataset is given in Table 4.3. For pair of domains from the Croatian dataset this results in 6985 features.

Domain pair	Features
Books – DVD	6559
Books – Kitchen	7559
Books – Electronics	7442
DVD – Kitchen	7608
DVD – Electronics	7445
Electronics - Kitchen	7032

Table 4.3: Number of features for each evaluated domain pair in the English dataset.

We establish two baselines, one for supervised and one for unsupervised domain adaptation. Both baselines are evaluated on a labeled set from target domain. Supervised baseline is referred to as Baseline+Target and is established by training a linear SVM using the labeled set from the source domain with addition of 100 randomly selected examples from the unlabeled set of the target domain. Frustratingly easy domain adaptation technique is compared to this baseline and is evaluated in the same manner. The only difference is that examples from the source and target domains are transformed using corresponding mappings as defined by FEDA. Unsupervised baseline is referred to simply as "baseline" and is established by training a linear SVM using only the source domain labeled set. Unsupervised domain adaptation techniques are evaluated similarly. When using mSDA and SCL for domain adaptation, feature mappings are determined using unlabeled sets from both domains and labeled set from source domain. Before training and evaluating the linear SVM, examples from both source and target labeled sets are transformed using learned mappings. DANN is trained using labeled set from the source domain and unlabeled set from both domains. DANN incorporates domain adaptation with classification, so it is also used for sentiment classification instead of

linear SVM. In case of KMM, instance weights for examples from source domain labeled set are computed using unlabeled set from the target domain. The computed weights are used in training of linear SVM on the source domain labeled set.

There are multiple hyperparameters to be tuned. Linear SVM has a hyperparameter C that is tuned using 5-fold cross-validation on the given training set. The hyperparameter C is selected among 10 logarithmically spaced values between 10^{-5} and 1. Domain adaptation technique's hyperparameters were tuned on the source domain labeled set. Specifically, the labeled source domain set is split into 10-folds. One fold is held aside for validation of the hyperparameters. Domain adaptation is done without the validation fold and the performance of the classifier trained on 4 remaining folds is evaluated on the validation fold. This is repeated two times with two different validation folds and average F_1 score is used as a criterion for selection of the best hyperparameters. mSDA has two hyperparameters: corruption probability of features and number of stacked denoising autoencoders. Corruption probability is either 0.25, 0.5, or 0.75, while number of layers is either 1, 3, or 5. SCL also has two hyperparameters: the number of pivots, which is either 500 or 1000, and the number of singular vector used for projection of new features, which is selected between values 50, 100, and 200. KMM is used with linear kernel which has no parameters. DANN has two hyperparameters: the domain regularization factor, selected from 9 logarithmically spaced values between 10^{-2} and 1, and the size of the hidden layer, which is either 50 or 100. A larger hidden layer as well as additional hidden layers lead to overfitting in preliminary experiments, so they were not considered. Also, DANN was optimized using Adam method for stochastic optimization (Kingma and Ba, 2014), instead of with standard gradient descent, since it converges more quickly. The learning rate was kept fixed at 0.001. For the rest of Adam optimizer parameters, the default TenforFlow values were used.

4.4. Evaluation Metrics

Performance of learned classifiers is evaluated using standard classification metrics: precision, recall, F_1 score, and accuracy (Alpaydin, 2014). These metrics are defined in terms of true positives (*tp*), true negatives (*tn*), false positives (*fp*), and false negatives (*fn*). True positives and true negatives are examples that are correctly classified as positive and negative, respectively. False positives are negative examples classified as positive, whereas false negatives are positive examples classified as negatives. Precision (P) measures proportion of correctly classified positive examples in set of all examples classified as positives:

$$P = \frac{tp}{tp + fp}. \tag{4.1}$$

Recall (R) measures the proportion of correctly classified positive examples in set of all positive examples:

$$R = \frac{tp}{tp + fn}. \tag{4.2}$$

F_1 score is defined as the harmonic mean between precision and recall:

$$F_1 = \frac{2PR}{P + R}. \tag{4.3}$$

Accuracy is defined as the proportion of correctly classified examples:

$$accuracy = \frac{tp + tn}{tp + fp + tn + fn}. \tag{4.4}$$

F_1 score is a more strict measure than accuracy since it takes into account precision and recall. Accuracy can be unrealistically high on imbalanced testing sets, e.g., Croatian dataset used in this thesis, whereas F_1 score reflects better the true classifier performance.

Simply comparing evaluation metrics between the baseline and a given domain adaptation technique is not sufficient to make conclusions about the performance of domain adaptation techniques since the difference between results might be due to chance. To this end, significance of difference between F_1 score achieved by a given domain adaptation technique and F_1 score achieved by the corresponding baseline is determined. Specifically, bootstrap resampling with replacement (Efron, 1979) is used. First, N new testing sets are created from the target domain labeled set. New testing sets have the same size as the target domain labeled set and are obtained by randomly sampling examples from the target domain labeled set (the same example can be sampled multiple times). The baseline and the given domain adaptation technique are evaluated on all new testing sets and differences between scores are obtained. We get an estimate of distribution of difference between baseline score and the given domain adaptation technique score from which we can derive a confidence interval for hypothesis testing with null hypothesis being that difference between scores is 0, i.e., baseline and domain adapted classifiers perform equally well. For example, if desired confidence interval is 95%, i.e., p-value lower than 0.05, differences between scores on randomly sampled testing sets are sorted, 2.5-th and 97.5-th percentiles are found, and these define the 95% confidence interval. If difference of scores between DA technique and baseline on original testing sets falls outside confidence interval, we reject the null hypothesis, i.e., the difference in difference in results is statistically significant with $p \leq 0.05$.

4.5. Evaluation Results

This section deals with evaluation results of domain adaptation techniques applied to the problem of sentiment classification. There were 12 domain adaptation tasks for the English dataset and two domain adaptation tasks for the Croatian dataset. The methodology

of experiments was given in section 4.3. Subsection 4.5.1 presents and discusses evaluation results for the English dataset, whereas subsection 4.5.2 presents and discusses results for the Croatian dataset.

4.5.1. Results for the English Dataset

Table 4.4 shows the evaluation results of the in-domain classifier. Results range from F_1 score of 0.828 for the DVD domain to F_1 score of 0.886 for the kitchen domain. In-domain results are gold standard which we hope to achieve with domain adaptation.

Domain	P	R	F_1	Acc
Books	0.854	0.857	0.855	0.856
DVD	0.833	0.821	0.828	0.829
Electronics	0.858	0.873	0.864	0.865
Kitchen	0.895	0.876	0.886	0.886

Table 4.4: Evaluation results of in-domain classifier for all domains.

Each of the four domains is used as target domain and domain adaptation performance is evaluated when adapting from each of the three remaining domains. Table 4.5, Table 4.6, Table 4.7, and Table 4.8 give evaluation results when adapting to DVD, books, kitchen, and electronics domains, respectively. F_1 scores statistically different between the baseline and the given domain adaptation technique at $p \leq 0.05$ are marked with a *.

Source		Baseline	SCL	KMM	mSDA	DANN	Baseline+Target	FEDA
Books	P	0.795	0.799	0.781	**0.839**	0.784	0.**820**	0.800
	R	0.774	0.773	0.775	0.803	**0.815**	0.754	**0.768**
	F_1	0.787	0.789	0.779*	**0.824***	0.795*	**0.794**	0.788*
	Acc	0.787	0.789	0.779	**0.825**	0.795	**0.794**	0.788
Kitchen	P	0.714	0.722	0.737	**0.746**	0.735	0.750	**0.771**
	R	0.714	0.744	0.722	**0.759**	0.725	**0.730**	0.689
	F_1	0.713	0.729	0.691	**0.786***	0.731*	0.692	**0.742***
	Acc	0.714	0.729	0.723	**0.759**	0.732	0.731	**0.742**
Electronics	P	0.719	0.718	0.691	**0.738**	0.728	0.749	**0.772**
	R	0.705	0.724	0.763	**0.766**	0.713	0.706	**0.736**
	F_1	0.714	0.719	0.710	**0.747***	0.727*	0.735	**0.759***
	Acc	0.715	0.719	0.711	**0.747**	0.728	0.735	**0.759**

Table 4.5: Evaluation results with DVD as the target domain.

Source		Baseline	SCL	KMM	mSDA	DANN	Baseline+Target	FEDA
DVD	P	0.786	0.793	0.784	**0.837**	0.776	**0.787**	0.779
	R	0.711	0.724	0.656	**0.817**	0.747	0.732	**0.738**
	F_1	0.758	0.767	0.736	**0.829***	0.765	**0.767**	0.764
	Acc	0.758	0.767	0.738	**0.829**	0.765	**0.767**	0.764
Kitchen	P	0.677	0.680	0.671	**0.751**	0.747	0.698	**0.717**
	R	0.739	0.748	**0.784**	0.781	0.586	**0.736**	0.709
	F_1	0.693	0.697	0.698	**0.761***	0.690	0.709	**0.715**
	Acc	0.694	0.698	0.699	**0.761**	0.694	0.709	**0.715**
Electronics	P	0.715	0.727	0.681	**0.806**	0.702	**0.713**	0.704
	R	0.665	0.662	**0.736**	0.674	0.689	0.714	**0.719**
	F_1	0.699	0.706	0.696	**0.754***	0.698	**0.711**	0.708
	Acc	0.699	0.707	0.692	**0.756**	0.698	**0.711**	0.708

Table 4.6: Evaluation results with books as the target domain.

Source		Baseline	SCL	KMM	mSDA	DANN	Baseline+Target	FEDA
Books	P	0.785	0.737	0.809	**0.892**	0.741	**0.801**	0.784
	R	0.702	0.714	0.697	0.708	**0.764**	0.748	**0.812**
	F_1	0.754	0.757	0.765	**0.809**	0.748	0.781	**0.794**
	Acc	0.755	0.757	0.767	**0.811***	0.749	0.781	**0.794**
DVD	P	0.798	0.813	0.795	**0.870**	0.786	**0.803**	0.789
	R	0.733	0.715	0.715	**0.785**	0.732	0.773	**0.805**
	F_1	0.774	0.774	0.765	**0.834***	0.766	**0.792**	0.789
	Acc	0.774	0.775	0.765	**0.834**	0.766	**0.792**	0.789
Electronics	P	0.822	0.827	0.822	0.811	**0.855**	**0.840**	0.810
	R	0.847	0.858	0.876	**0.898**	0.826	0.871	**0.878**
	F_1	0.831	0.839	0.843	**0.844**	0.843	**0.853**	0.836*
	Acc	0.832	0.839	0.843	**0.845**	0.843	**0.853**	0.836

Table 4.7: Evaluation results with kitchen as the target domain.

Source		Baseline	SCL	KMM	mSDA	DANN	Baseline+Target	FEDA
Books	P	0.788	0.801	0.808	**0.909**	0.757	**0.794**	0.779
	R	0.605	0.586	0.631	0.522	0.687	0.664	**0.766**
	F_1	0.717	0.715	**0.737***	0.722	0.733*	0.744	**0.774***
	Acc	0.721	0.721	**0.741**	0.735	0.734	0.746	**0.774**
Kitchen	P	**0.850**	0.827	0.846	0.846	0.828	**0.848**	0.824
	R	0.744	0.771	0.795	0.825	**0.839**	0.782	**0.796**
	F_1	0.806	0.805	0.825	**0.837***	0.833*	**0.821**	0.813
	Acc	0.806	0.805	0.825	**0.838**	0.833	**0.822**	0.813
DVD	P	0.808	0.815	0.831	**0.906**	0.781	**0.809**	0.785
	R	0.609	0.612	0.628	0.654	**0.679**	0.709	**0.758**
	F_1	0.729	0.732	0.745*	**0.789***	0.743*	0.770	**0.785**
	Acc	0.732	0.737	0.750	**0.793**	0.745	0.771	**0.785**

Table 4.8: Evaluation results with electronics as the target domain.

Comparing baseline results with in-domain results, we observe a degradation in performance. F_1 scores drop across of all source and target domain pairs, with the drop being smaller when adapting between similar domains, e.g., DVD-books and kitchen-electronics. For example, in-domain classifier for books domain achieves F score of 0.856. When adapting from DVD, the F_1 drops to 0.758, and when adapting from kitchen and electronics domains, it drops even more to around 0.69. This confirms that simply using the available source domain data to train a classifier for the target domain leads to poor performance, especially when source and target domains are different. Results for the Baseline+Target method indicate that using a small number of labeled examples from the target domain for training a classifier (in addition to labeled examples from the source domain) leads to better performance than when using unsupervised domain adaptation techniques, with the exception of mSDA. Using FEDA for supervised domain adaptation additionally improves performance in only 6 out of 12 domain adaptation tasks. SCL, KMM, and DANN show mixed performance, even worse than baseline in some domain adaptation tasks. SCL improves over baseline in 9 out of 2 tasks. On average, F_1 score is increased by only 0.005. Blitzer et al. (2007) experimented with the same dataset and also report that in some cases SCL can degrade the performance, but report bigger improvements over baseline for most of adaptation tasks. This might be due to a different experimental setup: they used all unigrams and bigrams found in unlabeled sets of given domain pair as features, whereas in this thesis we use only 5000 most frequent unigrams from both domains. KMM, on average, increases the F_1 score by 0.01 when compared to the baseline and also achieves the highest score of all domain adaptation methods when adapting from electronics to books. However, it improves the performance in only 6 of

12 domain adaptation tasks. To the best of our knowledge, no experiments using KMM for domain adaptation of sentiment classifier were reported in the literature. DANN improves the performance in 8 of 12 adaptation task when compared to the baseline. On average, the F_1 score is increased by 0.01 when compared to the baseline. This is in line with the results on the same dataset reported by Ajakan et al. (2014). mSDA turned out to be the best domain adaptation technique and consistently helps with adaptation across all domain adaptation tasks, in line with the results on the same dataset reported by Chen et al. (2012). The difference between mSDA and baseline F_1 scores is statistically significant for almost all domain adaptation tasks. On average, mSDA increases the F_1 score by 0.05 when compared to the baseline. Additionally, mSDA shows an improvement, for some domain adaptation tasks, when compared to supervised baseline (Baseline+Target).

With mSDA being the best performing domain adaptation technique, the viability of domain adaptation techniques for sentiment classification is discussed based on mSDA evaluation results. Table 4.9 summarizes F_1 scores for all 12 domain adaptation tasks with diagonal entries being F_1 score of in-domain classifiers (gold standard). mSDA performance

Source / Target	DVD	Books	Electronics	Kitchen
DVD	0.828	0.829	0.789	0.834
Books	0.824	0.855	0.722	0.809
Electronics	0.747	0.754	0.864	0.844
Kitchen	0.786	0.761	0.837	0.886

Table 4.9: F_1 scores for all 12 domain adaptation tasks using mSDA. Diagonal entries are F_1 scores of in-domain classifiers (gold standard).

gets close to gold standard when adapting between similar domains, i.e., books–DVD and kitchen–electronics. Although mSDA significantly improves performance in comparison against the baseline, mSDA F_1 scores are by a large margin lower than gold standard. The worst case is the adaptation from books to electronics domain where F_1 decreases by 0.14. Given target domain, mSDA might be sufficient for learning a classifier that performs almost as good as an in-domain classifier if source domain is similar to the target domain. However, if our source domain is not similar to the target domain, we can expect a drop in performance, albeit not as bad as if we ignore the domain adaptation problem.

4.5.2. Results for the Croatian Dataset

Table 4.10 shows the evaluation results of the in-domain classifier. The in-domain classifier for restaurant domain has an F_1 score 0.868, whereas for telecom domain it has an F_1 score of 0.707. Telecom domain is a harder problem in the context of sentiment classification since

it consists of social media comments that often violate the assumption that each document expresses the sentiment on single entity. On the other hand, restaurant domain consists of restaurant reviews for which this assumption holds. Large differences between F_1 scores and accuracies are due to imbalanced nature of the domains. Telecom domain has predominately negative documents, whereas restaurant domain contains mostly positive documents.

Domain	P	R	F_1	Acc
Restaurant	0.925	0.961	0.868	0.910
Telecom	0.735	0.681	0.707	0.908

Table 4.10: In-domain evaluation results.

Table 4.11 shows the evaluation results when adapting from restaurant domain to telecom domain. Baseline F_1 score is 0.276 lower than F_1 score of the in-domain classifier. Using labeled data from the telecom domain (Baseline+Target) results in the best F_1 score, comparable to gold standard, while additionally using FEDA outperforms the gold standard by 0.018. Using KMM results in performance worse than baseline, whereas SCL and DANN show small improvements when compared to baseline. mSDA performs best with F_1 of 0.675, which is an improvement over baseline score by 0.182. In contrast to other used unsupervised domain adaptation techniques, mSDA gets somewhat close to the performance of the in-domain classifier: F_1 score of the in-domain classifier is higher by 0.032.

Source		Baseline	SCL	KMM	mSDA	DANN	Baseline+Target	FEDA
Restaurants	P	0.240	0.250	0.227	**0.387**	0.243	0.429	**0.915**
	R	0.902	0.865	**0.926**	0.706	0.853	**0.769**	0.571
	F_1	0.493	0.519	0.458	**0.675***	0.508	0.709	**0.729***
	Acc	0.519	0.555	0.474	**0.770**	0.543	0.796	**0.848**

Table 4.11: Evaluation results with Telecom as target domain. DA techniques F_1 scores statistically different than baseline are marked with a *.

Table 4.12 shows the evaluation results when adapting from the telecom domain to the restaurant domain. Baseline F_1 score is only 0.383, which is lower than F_1 score of the in-domain classifier by 0.485. This is a considerably bigger drop in performance than is the case when adapting from the restaurant to the telecom domain. Using labeled data from restaurant domain beats unsupervised domain adaptation techniques and additionally using FEDA results in the lowest performance degradation; F_1 score drops by 0.123 in comparison with the gold standard. All unsupervised domain adaptation techniques improve F_1 scores over baseline: SCL by 0.02, KMM by 0.068, mSDA by 0.209, and DANN by 0.073. Although mSDA

achieves marked improvement over the baseline, its F_1 score of 0.592 is still considerably smaller than gold standard, which is 0.868.

Source		Baseline	SCL	KMM	mSDA	DANN	Baseline+Target	FEDA
Telecom	P	**0.964**	0.958	0.933	0.941	0.931	**0.916**	0.889
	R	0.208	0.236	0.309	**0.520**	0.454	0.807	**0.912**
	F_1	0.383	0.403	0.452*	**0.592***	0.456*	0.745	**0.777***
	Acc	0.386	0.405	0.452	**0.607**	0.556	0.796	**0.846**

Table 4.12: Evaluation results with Restaurants as target domain. DA techniques F_1 scores statistically different than baseline are marked with a *.

4.5.3. Discussion

The two evaluated domain adaptation tasks in Croatian language are more difficult than adaptation tasks in English language. The reason is that, unlike balanced domains in the English dataset, domains in the Croatian dataset are heavily imbalanced with telecom domain being mainly negative and restaurant domain mainly positive. In the first adaptation task, where restaurant domain is used for training a classifier to be used on telecom domain, the classifier gets biased towards positive class and consequently its performance on prevalently negative domain suffers: it will try to classify most of positive examples. Similarly, in the second adaptation task, the classifier gets biased towards thee negative class and is used on predominantly negative telecom domain. Class imbalance between source and target domains violates the covariate shift assumption that conditional source and target distributions are equal, $p_s(y|x) = p_t(y|x)$. We notice that unsupervised domain adaptation techniques, although making the covariate shift assumption, do improve over baseline. Additional observation is that using labeled examples from the target domain is more useful when the source and target domains are imbalanced. Using labeled examples from the target domain along, with or without FEDA, for two adaptation tasks with imbalanced domains in Croatian showed the best performance. On the other hand, the English dataset is balanced and using labeled examples from the target domain, with or without FEDA, was consistently outperformed by unsupervised domain adaptation techniques, i.e., mSDA.

5. Conclusion

One of the more popular tasks of sentiment analysis is sentiment classification and standard approach to tackle sentiment classification is to use supervised learning algorithms. A sentiment classifier trained using labeled data for one, source, domain will not perform as good on data from a different, target domain. Since opinionated user-generated text can belong to various different domains, a classifier trained on one domain would not be effective in some other domain of interest. Instead of labeling the data for each new domain of interest, domain adaptation (DA) techniques can be applied to make use of easily acquirable unlabeled data from the target domain.

The topic of this thesis was a study of domain adaptation techniques and their application for sentiment classification from text. DA techniques kernel mean matching (Gretton et al., 2009), structural correspondence learning (SCL) (Blitzer et al., 2007), frustratingly easy domain adaptation (FEDA) (Daumé III, 2007), marginalized stacked denoising autoencoders (mSDA) (Chen et al., 2012), and domain adversarial neural network (DANN) (Ajakan et al., 2014) were used for adapting sentiment classifier from the source to the target domain. Experimental evaluation was done in two languages: English and Croatian. The English dataset consists of four domains (12 DA tasks), whereas the Croatian dataset consists of two domains (2 DA tasks). Given the source and target domain, the mentioned DA techniques were applied using unlabeled data from both domains and the linear SVM was trained on the source domain and evaluated on the target domain. The results show that mSDA is the best performing domain adaptation technique. It outperforms all other techniques for all DA task in both English and Croatian.

Although mSDA improves the performance, it gets close to performance of the gold standard, i.e., the classifier trained on target domain data, only when source and target domains are similar,i.e., books and DVD reviews. Domain adaptation techniques do not match labeling sufficient amount data for the target domain in terms of classifier performance, but their application is cheap and effective in some cases. It would be interesting to see how other feature learning methods like mSDA, e.g., Paragraph Vector (Le and Mikolov, 2014) fare with differences between domains. Another interesting venue of research is using DANN in deep learning architectures to help with domain adaptation.

BIBLIOGRAPHY

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2014.

Shai Ben-David and John Blitzer. Analysis of representations for domain adaptation. *Advances in Neural Information Processing Systems*, 19:137–144, 2007. ISSN 10495258.

John Blitzer, John Blitzer, Mark Dredze, Mark Dredze, Fernando Pereira, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *Annual Meeting-Association for Computational Linguistics*, 45(1): 440, 2007. ISSN 0736587X. doi: 10.1109/IRPS.2011.5784441.

Julian Brooke. *A semantic approach to automated text sentiment analysis*. Doktorska disertacija, Simon Fraser University, 2009.

Minmin Chen, Kilian Q Weinberger, Fei Sha, and Los Angeles. Marginalized Denoising Autoencoders for Domain Adaptation. *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, stranice 767—-774, 2012. ISSN 0960-3174. doi: 10.1007/s11222-007-9033-z.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

Hal Daumé III. Frustratingly Easy Domain Adaptation. *Association for Computational Linguistic (ACL)s*, (June):256–263, 2007. ISSN 0736587X. doi: 10.1.1.110.2062.

Bradley Efron. Bootstrap methods: another look at the jackknife. *The annals of Statistics*, stranice 1–26, 1979.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

Goran Glavaš, Damir Korencic, and Jan Šnajder. Aspect-oriented opinion mining from user reviews in croatian. *ACL 2013*, stranica 18, 2013.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. U *Proceedings of the 28th international conference on machine learning (ICML-11)*, stranice 513–520, 2011.

Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009.

Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. Correcting sample selection bias by unlabeled data. U *Advances in neural information processing systems*, stranice 601–608, 2007.

Jing Jiang. A literature survey on domain adaptation of statistical classifiers. *URL: http://sifaka. cs. uiuc. edu/jiang4/domainadaptation/survey*, 3, 2008.

Yoon Kim. Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, stranice 1746–1751, 2014. ISSN 10709908. doi: 10.1109/LSP.2014.2325781. URL `http://emnlp2014.org/papers/pdf/EMNLP2014181.pdf`.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Qv Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. *International Conference on Machine Learning - ICML 2014*, 32:1188–1196, 2014. ISSN 10495258. doi: 10.1145/2740908.2742760. URL `http://arxiv.org/abs/1405.4053`.

Bing Liu and Lei Zhang. A survey of opinion mining and sentiment analysis. In Mining text data. *Springer*, stranice 415–463, 2012.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, stranice 1–12, 2013. ISSN 15324435. doi: 10.1162/153244303322533223. URL `http://arxiv.org/pdf/1301.3781v3.pdf`.

Rodrigo Moraes, JoãO Francisco Valiati, and Wilson P GaviãO Neto. Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Systems with Applications*, 40(2):621–633, 2013.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(12):1–135, 2008.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. U *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, stranice 79–86. Association for Computational Linguistics, 2002.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12 (Oct):2825–2830, 2011.

Leon Rotim and Jan Šnajder. Comparison of short-text sentiment analysis methods for croatian. *BSNLP 2017*, stranica 69, 2017.

DE Rumelhard, GE Hinton, and RJ Williams. Learning internal representation by error backpropagation, parallel distributed processing: explorations microstructure of cognition, vol. 1, 1986.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. Recursive deep models for semantic compositionality over a sentiment treebank. U *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, svezak 1631, stranica 1642, 2013.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307, 2011.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. U *ACL (1)*, stranice 1556–1566, 2015. ISBN 978-1-941643-72-3. doi: 10.1515/popets-2015-0023. URL `http://aclweb.org/anthology/P/P15/`.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. U *Proceedings of the 48th annual meeting of the association for computational linguistics*, stranice 384–394. Association for Computational Linguistics, 2010.

Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, svezak 1. Wiley New York, 1998.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

Sida Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. U *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, stranice 90–94. Association for Computational Linguistics, 2012.

Željko Agić, Nikola Ljubeić, and Danijela Merkler. Lemmatization and morphosyntactic tagging of croatian and serbian. 2013.

Domain Adaptation for Sentiment Analysis from Text

Abstract

Typical sentiment analysis systems are based on supervised classification algorithms which, in order to work well, assume that documents used for training of a classifier come from the same domain as documents sentiment of which classifier will be used to predict. In order to alleviate the need for labeling documents for each new domain of interest, domain adaptation techniques can be used with already available documents from some different domain. This thesis studies several domain adaptation techniques based on instance weighting and change of representation. The studied techniques were used for building domain-adapted sentiment classifierd based on linear support vector machine. The performance of domain-adapted classifiers was experimentally evaluated on texts in Croatian and English languages. Using domain adaptation technique based on marginalized denoising stacked autoencoders gave the best results across all domain adaptation tasks in both languages.

Keywords: Natural language processing, sentiment analysis, domain adaptation, cross-domain sentiment analysis, Croatian language, English language.

Postupci prilagodbe domeni za analizu sentimenta u tekstu

Sažetak

Uobičajeni pristup izgradnji sustava za analizu sentimenta temelji se na klasifikaciji sentimenta korištenjem algoritama nadziranog strojnog učenja. Da bi takav sustav radio dobro, dokumenti, čiji sentiment će naučeni klasifikator predviđati, trebali bi biti iz iste domene kao i dokumenti na kojima je klasifikator učen. Umjesto skupog označavanja dokumenata za svaku novu domenu, moguće je iskoristiti postojeće označene dokumente iz neke druge domene i primijeniti neki postupak prilagodbe domeni. U ovom radu proučeni su postupci prilagodbe domeni temeljeni na dodjeljivanju težina (engl. *instance weighting*) i postupci temeljeni na promjeni prostora značajki (engl. *change of representation*). Proučeni postupci iskorišteni su za prilagodbu domene pri učenju klasifikatora sentimenta temeljenog na stroju potpornih vektora. Postupci prilagodbe domene eksperimentalno su vrednovani pri izgradnji klasifikatora sentimenta za hrvatski i engleski jezik. Kao najbolji postupak za prilagodbu domene pokazala se primjena marginaliziranih odšumljavajućih naslaganih autoenkodera (engl. *marginalized denoising stacked autoencoders*).

Ključne riječi: Obrada prirodnog jezika, analiza sentimenta, prilagodba domeni, klasifikacija sentimenta, hrvatski jezik, engleski jezik.