Abstract

This project deals with approaches that automatically extract opinions from plain text a process known as *Opinion Extraction*. Supervised machine learning techniques have been used in order to extract opinions and compare reviews of research papers submitted to KDD – 2009 and ECML PKDD– 2012, two major international conferences of Machine Learning.

The project is divided in two parts that were parallel developed. In the first part, machine learning algorithms have been trained using review texts and scores in order to build models that solve three general learning problems: classification, regression and ranking. Classification categorizes reviews (negative, neutral, positive) based on the opinions expressed on the review text, regression refers to an alternative way of predicting numerical scores or in this case review score ratings and ranking was used to rank the reviews from those that express the most positive opinion to those that express the most negative opinion.

The second part investigates whether the relationship between review text and review scores of the two conferences are different. In order to do that, the models trained during the first part were applied from one conference to the other, a process known as *transfer learning*. From this process which allowed me to compare the two conferences, many similarities and differences have been revealed based on the performance of the models. Furthermore, alternative techniques have been evaluated regarding the models that could not been applied to different domains.

This is a research-style project (Type II). The purpose of this project was to combine existing methods proposed by different researchers in order to see if they can be adopted for solving various sentiment learning problems in this particular corpus. Many experiments have been performed, that were aiming not to solve all of the limitations but to understand and learn different problems, evaluate the results, reveal limitations and strong points and extract specific information about the way that authors express their opinions when they review research papers. The main contributions are as follow:

- This study is one of the first to compare reviews from different conferences using machine learning techniques. Experiments were carried out that extract the most frequent features between the two conferences, similarities and differences between the two conferences, models that solve different opinion mining problems.
- I proposed a learning framework that combines existing approaches and can be used in order to extract opinions from reviews of research papers and also apply transfer learning between different conferences.
- I use state-of-the-art machine learning algorithms in combination with feature extraction methods that can capture opinions. I evaluate the outcome and recommend the best models per learning problem and conference and also the reasons that some models succeed or fail.
- This project not only solves but also transfers many different learning problems in just a single work. The three learning problems have been compared and the analysis of the results shows which is easier to be transferred and which is easier to be learned.
- Finally, for models that could not be transferred I experimented with an alternative measure that extracts generalized features with a cleverer way, as proposed by researchers in *Transfer Learning* area, which shows that methods in this field would be useful in order to transfer knowledge between conferences.

Table of Contents

List of Figures

List of Tables

1 Introduction

Opinion mining or *Sentiment analysis* (Pang & Lee 2006a) focuses on techniques that detect opinions in documents, which has been a very popular research area during the last few years. The main reason arises from the fact that huge amounts of opinionated documents such as reviews, discussions in forums or in twitter are available and opinions, sentiments or judgments can be useful for various different reasons. Human beings are interested in opinions of other people especially in cases they want to take decisions. Companies can take advantage from the huge amount of product reviews and adapt a particular management policy. Opinion extraction from reviews of research papers gives the opportunity to investigate how the language of reviewers corresponds to their ratings and if they are consistent to the way they score different papers based on their review text. *Opinion mining* searches for ways that can deal efficiently with the large amount of opinionated documents without the need of human's supervision. This field is able to replace conventional methods such as focus groups, surveys and opinion polls.

Opinion extraction (Tang et al. 2009) is the process of automatically extracting opinions and it is highly related with the field of Natural language processing. This information gives the opportunity:

- To classify reviews. For example, as positive, negative or neutral, based on the opinions/judgments expressed on the review text. This process is known as sentiment classification on document level.
- To predict numerical scores. Reviews come with a score range or a star range and methods have been developed that predict these scores based on the review text.
- To rank them. Rankers output a preference rank where reviews that are higher in the rank are those that have positive polarity towards a theme and those that are lower have negative polarity.

One dimension along which we could categorize opinion extraction methods is based on the training data. If a method require both the review text and the review score, which is used as a ground truth, this is called Supervised learning and if the review score is not taken into consideration this is called Unsupervised learning. For this project we focus on supervised learning methods where machine learning algorithms are used.

The majority of machine learning algorithms are able to achieve high performance only when the training set and test set are derived from the same distribution. There are cases that we are able to learn one domain, for example product reviews and we also want to apply what we have learned to a different domain like movie reviews. However, in many real applications it is impossible to retrain a classifier whenever the distribution of data changes. *Transfer learning* is the field that investigates different approaches in order to transfer knowledge between different domains without rebuilding a model from scratch. Sentiment classification has been predicted to be very sensitive to a specific domain. As a result, using the same model between different corpora leads to poor results.

Although different kinds of opinionated document have differences in vocabulary and structure, the general process and methods of automatically extracting opinions are very similar. However, for this project I have mainly focused on methods applied to reviews. Reviews of research papers submitted to two different international conferences were available and existing methods have been adopted in order to extract opinions and compare them. The most of researchers have focused on product and movie reviews as they have more practical interest and can easily been obtained. This study is among the first that combines existing methods in this different kind of reviews.

The rest of this section underlines the aims and objectives of this project and also the outline of the dissertation.

1.1 Aims and objectives

A project has been developed that focuses on opinion extraction from reviews of research papers submitted to two different major international conferences. The first conference is the KDD - 2009[1] and the second conference is the ECML - 2012[2]. International conferences have a common process which deals with the acceptance or rejection of scientific papers. I briefly explain this process here and the reader can refer to section 4.1, where the complete process is extensively described. According to peer review process, authors send their scientific papers to conferences. Reviewers read them and express their opinion, at the same time they give a score which shows their willingness to reject or accept the paper.

The two conferences have a different score rating. Reviews in KDD could choose from a six score range, while reviewers in ECML could choose from a four score range. Having the scores from each review and using them as the ground truth gives the opportunity to use supervised machine learning approaches in order to analyze them. The aims and objectives of this project are as follows:

Aim: The first aim of this project is to use supervised learning in order to classify, predict numerical scores and rank the reviews submitted to the two conferences based on the opinions that the reviewers express on the review text.

Objectives:

- Create a learning framework in order to solve opinion mining learning problems and develop a common process to evaluate the different approaches.
- Extract those features that are more likely to capture opinions.
- Use supervised learning and train models in order to solve different learning problems: binary classification (distinguish reviews that reveal positive opinion from the negative reviews), multi-class classification (strong negative, negative, positive, strong positive), regression (predict review scores) and ranking (rank reviews from those that express positive opinion to those that express negative opinion).
- Investigate how difficult it is to learn each of these problems and why it is difficult.
- Find the best combination of features and supervised learning algorithms that leads to the best possible performance.

Aim: The second aim is to investigate whether the relationship between review text and review scores of these two conferences are different.

Objectives:

- Extend the learning framework in order to support transfer learning.
- Apply the models trained on the first conference to the second conference and vice versa.
- Investigate which models and which of the three learning problems (classification, regression and ranking) can easier be transferred.
- Search for alternative techniques in order to achieve a better performance.

[1] Knowledge Discovery and Data Mining (KDD)
[2] European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)

1.2 Outline

This dissertation consists of eight sections that are organized as follows:

Section 2 offers the appropriate background knowledge, which describes the *Opinion mining* area, the opinion mining problems that need solution and the methods that the researchers have developed in order to solve them.

Section 3 reviews related work regarding the Transfer learning problem and the reasons that traditional machine learning fails.

Section 4 introduces the reader to the various opinion mining problems that this project aims to solve and transfer, the datasets that have been analyzed, the assumptions behind the experimental strategy and the challenges of this project.

Section 5 provides an overview of the software, the machine learning algorithms and the feature extraction methods that have been used. It also presents the proposed learning framework that combines all of these in order to achieve the scope of this project. Finally, it describes the complete evaluation process and the statistical tests performed during the experimental phase.

Section 6 presents the empirical study. In this section many experiments have been carried out. The experimental results are presented and analyzed, which among others lead to recommendations about the best ways of extracting opinions from reviews of research papers.

Section 7 provides a discussion on the results and a summarization of the outcomes. It also evaluates this project by noting in what extent aims and objectives have finally been completed.

Section 8 presents conclusions and recommendations for future work.

2 Opinion Extraction and Sentiment Analysis

This section and the next one are dedicated to background material. I explain the areas related to this project, the way they are connected with each other and why they are important regarding the project.

2.1 Overview and definitions

Sentiment analysis or *opinion mining* (Liu 2010; Pang & Lee 2006; Tang et al. 2009) is described in the bibliography by having almost the same meaning. Sentiment can be expressed as judgment (opinions, view), affect (emotion, feeling) or attitude. For this project we focus on opinions so a general definition could describe this area as the one that searches for opinions into words, sentences and documents.

Product reviews, movie reviews, discussions in forums or in twitter are some examples of opinionated documents. In order to formalize the problem we will refer to any set of these documents as the training data. Review documents normally come with a score, which is assigned by the author and corresponds to his preference towards the theme he reviews. For example, it can be the star rating for product reviews or a score rating (1 to 6) for reviews of research papers. Approaches of the field use this score as the ground truth. Of course, opinionated texts do not always come with a score and methods have been developed that can deal with unlabeled data.

Opinion extraction is the process of automatically extracting opinions from plain text. One dimension along which we could categorize opinion extraction methods is based on the type of the data. When labeled data are required this is called *Supervised learning*. Methods of this category use machine learning algorithms. Methods that deal with unlabeled data are known as *Semantic orientation* or *Unsupervised learning* and they use either machine learning algorithms, for example clustering algorithms, or they are based on other techniques that extract the sentiment polarity or opinion orientation of words into a document (Zhou & Chaovalit 2005).

Another dimension along which we could categorize opinion extraction methods is based on the learning problem they aim to solve. Opinion extraction can be used for classification, numerical score prediction or ranking. In the next section, I describe the mathematical properties of each learning problem that reveal why different approaches are more suitable for different learning problems.

2.2 Learning problems

The field of *Machine Learning* is referred to classification as the task of classifying an object to one or more categories, to regression as the task of predicting a real value for an object and to ranking as the task of ordering a number of objects starting from the most preferable to the least preferable (Witten & Frank 2005). This section shows how these problems are interpreted by the Sentiment Analysis area. Let $\{\vec{x_i}, y_i\}$, $i = 1 \ldots m$ denote a set of m reviews or training examples, where each input vector $\vec{x_i}$ is a different review while $y_i \in \{c_0, .., c_n\}$ where $c_0 < c_1 < \cdots < c_n$ is the value of the ordinal target variable, in other words the score that the reviewer gives to review i. Researchers use machine learning in order to extract opinions and solve the upper mentioned learning problems.

Classification based on the sentiment is known as *Sentiment classification* or *Polarity classification* and can be further separated into binary classification and multi-class classification. Furthermore, Sentiment classification can be achieved in three different levels: Sentence-level classification, word-level classification and document-level classification.

Different approaches have been developed for each of these levels. However, even if these are studied as three different areas, advances in word and sentence level classification are used for document-level classification (Tang et al. 2009). Assuming that $y_i \in \{c_0 = negative, c_1 = positive\}$ then binary sentiment classification is to classify every review i to one of these two categories. On the other hand, if there are more than two categories like $y_i \in \{c_0 = negative, c_2 = neutral, c_3 = positive\}$ then multi-class sentiment classification is to be used. As far as the number of classes increases, classification becomes a very difficult problem. For example, predicting the score for each review can be treated as a classification problem if the initial score range is known, however, regression is a more sensible solution.

Researchers use regression schemes in order to predict review ratings for example the number of stars that the reviewer gives to his review (Pang & Lee 2006b, Zhang & Varadarajan 2006, Baccianella et al. 2009b). Regression schemes assume that $y_i \in \mathbb{R}$ which is less restrictive compared to the assumption of classification schemes.

Ranking a set of reviews can be enough for particular applications. In this case, we are interested in extracting a list of shorted reviews from those with positive polarity to those with negative polarity. Rating scores give exactly this information however, classification algorithms do not take into account that the class attribute is ordinal. Probably, learning a ranking problem can be easier than learning a multi-class or regression problem. *Preference learning* is the broader area that focuses on this kind of ranking. More specifically, the methods that learn to rank instances are known as *instance ranking* methods in order to distinguish them from *label ranking* and *object ranking* methods where in the first case for every instance all of the distinct values in the class attributes have to be shorted and in the latter unsupervised learning to rank approaches have been developed in order to deal with unlabeled training data (Fürnkranz & Hullermeier 2011: 3-7). Sentiment analysis field has mainly focused on regression and classification schemes but ranking is also required by some applications.

To sum up, both regression and classification schemes train a function f with f($\overrightarrow{x_i}$) = y_i . Classification approaches try to map $\overrightarrow{x_i} \rightarrow \{c_0, .., c_n\}$ and regression approaches try to map $\overrightarrow{x_i} \rightarrow \mathbb{R}$. Ranking methods take into account that class label indicates a preference (c_4 is more preferable than c_1) and given a set of reviews with score values according to this: $y_1 > y_2 > y_5$, they return scores where f($\overrightarrow{x_1}$) > f($\overrightarrow{x_2}$) > f($\overrightarrow{x_5}$) which indicate that $\overrightarrow{x_1} \succ \overrightarrow{x_2} \succ \overrightarrow{x_5}$. The symbol $\succ$ implies preference (Duh 2008). The way that ranking methods are evaluated is also different compared to classification and regression methods. The simplest way to evaluate a ranker on the test set is to consider a ranking error as the one where the ranker has rank $\overrightarrow{x_i} \succ \overrightarrow{x_j}$ while $y_i < y_j$. So, the number of ranking errors is the number of pairs that have been ranked wrongly. Regression methods can be evaluated by calculating the mean square error which is the distance between the true label and the label that the regression function assigns. Finally, classification can be evaluated by counting the number of correct and incorrect classifications which is known as the error rate. As an example, assume that a model has been trained on reviews from research papers where score ratings go from 1(strong accept) to 6(strong reject). Now, we have three new reviews, the trained model assigns 3, 4, 5 and the real scores are 2, 3, 4. As a multi-class classification problem the classifier makes 3 errors. As a regression problem the mean square error is equal to 1 and if the trained model is a ranker then there is not error as the three new reviews have been shorted perfect. This example illustrates that a ranker learns the relative order and not specific numbers. This is a different point of view that probably is easier to be solved.

2.3 Feature extraction and vector representation

Different approaches have been used in order to solve the sentiment learning problems that have been described in the previous section. The first step for the most of them is to convert the plain text into a form which is meaningful and recognizable by the learning framework. As a result, the most of the researchers follow a very similar preparation phase where linguistic analysis processes are taking place as presented in Figure 1. The output of this process is a form, which is known as feature vectors. Reviews are run through a parser, features are extracted and then each review is represented by a vector.

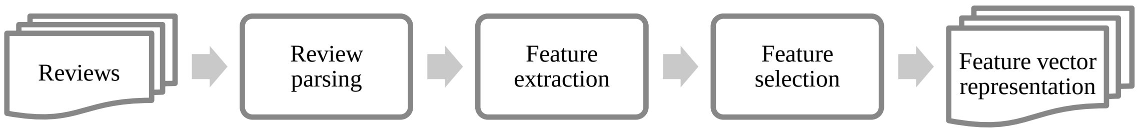


Figure 1 Feature vector construction.

2.3.1 Review Parsing

Starting with a raw document some preprocessing techniques are applied to it, for example it is separated into sentences and these sentences are run through a parser, they are further split in word-tokens and the parser can tag them with more information like the parts of speech.

2.3.2 Feature extraction

Natural language documents are regarded as unstructured data when they are given to a machine. The aim of feature extraction is to give a structure to the text, in order to be understood by a machine and also to discover those features that are able to detect opinions. Different feature extraction approaches have been proposed by different researchers, the most common of them that have been predicted to capture opinions are presented in this section. A number of them have also been used during the experiment section regarding this project.

Terms and bag-of-words: These features are individual words or word n-grams. N-grams are two or more words (n indicates the number) that are combined together. Word positions may also be considered (Pang et al. 2002). This method results to a vector of different words which is known in bibliography as a bag-of-words vector and many different approaches have been proposed in order to represent it. The most of researchers experimented only with unigrams. Pang & Lee (2004) and Pang et al., (2002) tried to extract bigrams, however they could not achieve better results. On the other hand, (Dave et al. 2003), Ng et al., (2006) and Kennedy & Inkpen (2006) found that specific bigrams or trigrams improve the performance over just extracting unigrams.

Part Of Speech tags (POS): Many researchers maintain that particular parts of speech such as adjectives and adverbs reveal opinions. A more advanced method has been proposed by Turney (2002) where pairs of adjacent words and not just single words are extracted in cases that they follow specific patterns as described in Table 1. For example, the second raw says: Extract two words if the first one is an adverb (RB) or adverb comparative (RBR) or adverb superlative (RBS), the second one is an adjective (JJ) but the third one is not a noun singular (NN) or plural (NNS). Ng et al., (2006) note that the most of researchers focus on JJ-NN pattern. Reader can refer to Santorini (1995) who gives a complete description for all of these tags.

First word	Second word	Third word (Not Extracted)
1. JJ	NN or NNS JJ	anything
2. RB, RBR, or RBS	JJ	not NN nor NNS
3. JJ	JJ	not NN nor NNS
4. NN or NNS	JJ	not NN nor NNS
5. RB, RBR, or RBS	VB, VBD, VBN, or VBG	anything

Table 1 Patterns of POS tags for extracting two-word phrases (taken from Turney(2002))

Opinion words and phrases (also known as *polar words*, *opinion-bearing words*, and *sentiment words*): Opinion words and phrases are those that are known to reveal positive (nice, good, well-developed) or negative opinions (bad, poor etc). It is obvious that these words are important indicators of polarity orientation. In order to extract this information opinion lexicons can be used. Generally, opinion lexicons include terms with their polarities. For example, SentiWordNet lexicon (Esuli & Sebastiani 2006) offers a variety of terms where each of them is associated with three different values: Obj(s) which indicates how much objective is the word s, Pos(s) which indicates if the word s is positive and Neg(s) which indicates if word s is negative. The main drawback with lexicon-based approaches is the fact that a word which is included in an opinion lexicon does not always express opinion into a particular sentence (Liu 2010). However, the combination of opinion lexicons with machine learning algorithms is a trend that seems to work quiet well. Ohana & Tierney (2009) take advantage of SentiWordNet lexicon in order to extract the semantic orientation of terms that exist in each review and train supervised machine learning algorithms. Mingzhi et al. (2009) build their own polarity word-list based on positive and negative documents. They use a graph-based approach in order to represent the documents and the page-rank algorithm to give polarity scores to every word. Their approach is able to take advantage from the context of the sentence and it outperforms robust methods such as those proposed by Pang et al. (2002) and Turney (2002) (these methods are discussed later).

Syntactic dependency: Words dependency based features generated from parsing or dependency trees have also been tried by several researchers. For example, Ng et al. (2006) use the MINIPAR dependency parser in order to extract subject-verb and verb-object relations. However, this information was not able to give better results compared to the accuracy achieved by extracting only n-grams.

Valence shifters: Negations, overstatements and understatements are some categories of valence shifters (Kennedy & Inkpen 2006, Ohana & Tierney 2009). Negation words change the opinion orientation of a phrase, overstatements and understatements are words that increase or decrease the strength of a term, respectively. For example, the word “very” in a phrase such as “very good” is an overstatement because it increases the strength of term good.

2.3.3 Feature vector representation

Each review can be represented by a raw vector, which is known as feature vector. A number of different ways in order to represent a raw vector of extracted features have been proposed. The most common in *sentiment analysis* area are frequency counts and Boolean values. In the first case, each feature is represented by the times it occurs into the document and this number probably has to be normalized in order to be fair for documents that have different lengths. In the second case, each feature takes a Boolean value (1 or 0). 1 indicates that the feature appears into the document 0 indicates that the feature does not appear into the document. Pang et al. (2002) and Kennedy & Inkpen, (2006) concluded that feature presence lead to better results compared to feature frequency for particular algorithms which is opposite as they maintain to other works on topic-based classification. On the other hand, Baccianella, Esuli, & Sebastiani, (2009) applied the TF x IDF (Term Frequency times inverse document frequency) weighting scheme from information retrieval which also has predicted to be very robust.

These are the three most commonly used approaches in sentiment analysis. Many more different ways have been proposed. Some of them are described by Dave et al. (2003) who experimented with a number of techniques from information theory such as: Information gain, Fischer discriminant, counting, Jaccard's measure of similarity etc. The choice of the correct scheme can be decided only after experiments because it depends on the problem and the learning schemes that will be selected to solve it.

2.3.4 Feature selection

Basic preprocessing techniques and feature extraction methods such as POS tagging and syntactic dependencies are able to reduce the feature space. However, in many cases there is a need to reduce even more the feature space in order to be easier for the learning algorithms to handle it. Feature selection or feature reduction is a very important stage for solving almost any learning problem. Different methods are used in order to reduce the feature space and hold those features that can give as much information as possible. The most commonly used approaches in sentiment analysis are Mutual Information (Wen Fan et al. 2011) and Information Gain. Information Gain shows the discriminating power of every feature which is the amount of information that a particular feature can give for a specific class. The entropy for a binary classification is defined as:

$$Entropy(subset) = -(P(C^{+})logP(C^{+}) + P(C^{-})logP(C^{-}))$$

The information gain for a word is defined as:

$$Gain(term) = P(term)Entropy(subsetIT) + P(!\,term)Entropy(subsetNT)$$

In order to compute the information gain of a term we can divide the dataset to two subsets the one subset includes all of the documents that have this term (subsetIT) and the second all of the documents that do not have it (subsetNT). High values for entropy indicate that the term exist equal times for both negative and positive class, lower values indicate that this term is possibly discriminating. The terms are ordered according to their information gain and the k-highest ranking are selected.

Other approaches have also been proposed for example Ng et al., (2006) and Aue & Gamon (2005) select the k-highest ranking features according to WLLR weighted log-likelihood ratio. While Baccianella et al., (2009) argue that the most of the approaches focus on classification so they propose two feature selection methods suitable for ordinal regression models named Minimum Variance and Round Robin Minimum Variance. The reader can refer to the corresponding papers for more details.

2.4 Supervised methods

The most robust methods for *Sentiment analysis* and *Opinion extraction* are supervised learning approaches, which use machine learning algorithms and work with labeled reviews. Supposing we have a corpus of review documents during the training phase a machine learning algorithm is trained in order to build a model and during the test phase new unlabeled reviews are classified by this model. The most competitive approaches, as proposed by other researchers, are presented in this section. APPENDIX A includes more details for those algorithms that have also been used in the experimental section.

2.4.1 Machine learning algorithms for classification

Many different supervised machine learning algorithms have been proposed for sentiment classification in document level. Researchers have mainly focused on binary classification where the most commonly used classifiers known to achieve the highest accuracy are Naïve Bayes (NB), Multinomial Naïve Bayes (MNB) and Support Vector Machines (SVMs). Other schemes have also been used like Decision trees, Expectation Maximization (EM) etc.

Pang et al. (2002) are the first to use supervised machine learning methods for opinion extraction and binary classification (predict whether a review is positive or negative). They created a movie review corpus, which has also been used by many researchers, in order to test their approaches. In their experiments SVMs outperform NB and EM. The authors experimented with many different feature extraction methods such as unigrams, bigrams, part of speech, position and negation tagging; however, using only unigrams was found to give the best results. Recently a different approach proposed by Lin, Tan, & Cheng, (2011). Their method consists of two steps. In the first step, they distinguish “key sentences” from “trivial sentences”. Key sentences are those that reveal a higher degree of opinions and they are extracted based on the position of each sentence, the fraction of opinion words and some hand-selected phrases that usually appear in key sentences. In the second step, an ensemble of three classifiers is trained on key and trivial sentences. As far as I am concerned this is the only sentiment classification approach which takes advantage of ensemble methods.

Multi-class classification is a more difficult problem where classifiers present lower performance. Pang & Lee, (2005) experimented with SVM one-versus-all implementation which is an extension of the standard SVM that can be used for multi-class classification.

2.4.2 Machine learning algorithms for review score prediction

Not only classification but also regression schemes have been recommended in order to predict reviewing scores. Pang & Lee, (2005) compared SVM classifier with ε-SVR (support vector regression), which is a regression version of SVM algorithm. They concluded that for three classes a classifier performs better than a regression model. However, as far as the number of classes increases a regression model should be preferred. ε- SVR has also been used by Baccianella et al. (2009) in order to predict review ratings. Finally, Z. Zhang & Varadarajan, (2006) compared ε-SVR with linear regression and concluded that the first outperforms the second.

2.4.3 Machine learning algorithms for ranking

Probabilistic classifiers and regression models can be used as rankers. However, a complete area , introduced in section 2.2, deals with methods that exactly take advantage of the way that ranking schemes are evaluated thus advanced ranking schemes have been developed that learn preferences between objects and not exact scores or classes. Sentiment analysis field has mainly focused on regression and classification schemes while ranking has been investigated as a different problem by other fields like *document retrieval*. However, there are sentiment analysis applications where just knowing the order of the reviews from the most positive to the most negative is enough. In this section I present two ways that can be used in order to automatically rank a set of reviews:

- Classifiers and regression models as rankers (Lachiche & Flach 2003): Regression models output a numerical prediction regarding each test instance. This prediction can be used in order to rank the data. On the other hand, probabilistic classifiers like MNB output posterior probabilities over the classes. The output can be converted to a score which can be used in order to treat the classifier as a ranker. As proposed by

(Lachiche & Flach 2003) probabilities can be converted into scores using the following formula:

$$\text{score} = \frac{f(Pos, review)}{f(Neg, review)} = \frac{f(Pos, review)}{1 - f(Pos, review)}$$

Where $f(Pos, review)$ ($f(Neg, review)$) corresponds to the probability of this review to be positive (negative). Of course, this formula can be applied only to classifiers that have been trained in order to solve binary classification problems.

- Approaches for *Learning To Rank* problems: RankNet is a pair wise ranking method, more specifically it is a neural network ranking model proposed by Burges et al. (2005) which has been used for document retrieval by search engines. This is a probabilistic method which uses neural networks as models and gradient descent as the algorithm in order to optimise a rank loss function. A set of reviews including their scores is given to the ranker; it is trained and produces a model which can be used in order to rank new unlabeled reviews.

2.5 Unsupervised and Semi-supervised methods

Alternative approaches use unlabeled data and they are known as unsupervised learning approaches. They have been predicted less robust than supervised approaches and they have not been used in this project, however, some interesting of them are presented in this section. Finally, recently the most competitive approaches are the semi-supervised where a combination of labeled and unlabeled data is used.

One of the first unsupervised approaches has been described by Turney (2002). This paper comprises a source of inspiration for a future evolution in this area. A review is classified based on the polarity of the majority of the phrases. If the majority of the phrases in this review are positive then the review is classified as recommended (positive) otherwise it is classified as no-recommended (negative). AltaVista search engine is used in order to count the number of hits for each of the phrases that have been extracted from the review. A representative approach regarding the semi supervised category is the one proposed by Dasgupta & V. Ng (2009), where many different fields such as spectral clustering, active learning, transductive learning and ensemble methods are combined. This results to a state-of-the-art approach that leads to high accuracy for five different domains. However, we will not expand more on this because it is out of the scope of this work

2.6 Summary

Sentiment classification and opinion extraction is a modern machine learning and text mining problem, with very different properties than topic-based classification problems. Many researchers have developed different techniques in order to extract opinions from reviews and solve learning problems like classification, review score prediction and ranking. Supervised and unsupervised learning approaches have been proposed for solving them and recently a combination of these two has also lead to semi-supervised learning approaches. For this project, we are more interested in supervised learning approaches, so the focus was there. SVMs and NB algorithms have been used by the most of researchers and seem to work greatly for this field. Many research papers underline the importance of a strong feature space and the fact that every machine learning algorithm is as good as the feature space is. For this reason, different feature extraction methods that are more likely to capture opinions have also been presented in this section. Little research has performed in combination of more classifiers and probably ensembles of classifiers would give better results in this field.

3 Compare conferences, Transfer Learning

The second aim of this project is to compare the two conferences. In order to do that models are trained on the first conference, they are applied to the second conference and vice versa. This process is known as *transfer learning* because it gives the opportunity to learn a second domain without rebuilding models from scratch. The challenge here is that Machine learning algorithms are very domain sensitive. If the language in the second conference is different or the score ratings are different, then it is very possible that the models built from the first data set will perform poor. Poor performance in our case indicates that the two conferences have differences, which is enough regarding this project. However, if one is interested in improving the performance across domains then transfer learning techniques should be used that try to solve these limitations of traditional machine learning methods. Different approaches have been developed for classification, regression, clustering problems and reinforcement learning. For this project I do not use transfer learning algorithms, however, a brief description of this area and the reason that traditional machine learning fails or succeeds is useful, in order to explain the performance of machine learning schemes and understand if transfer learning area would be useful for the specific datasets.

3.1 Overview and definitions

Traditional machine learning approaches make a basic assumption in order to build statistical models: training data and test data come from the same distribution. So, in the case that the test data come from a different distribution they perform poor. Transfer learning (also known as *learning to learn, life-long learning, knowledge transfer)* methods are here to deal with this limitation. According to the definition that Pan & Qiang Yang (2010) have given, every transfer learning problem consists of two *domains* with their *learning tasks*: the source domain Ds with a learning task Ts, and the target domain Dt with a learning task Tt. Transfer learning aims to improve the learning in Dt by using the knowledge in Ds. The learning task is to find an approximation function based on some label data. A domain is a set of training examples with their labels.

In order to use transfer learning, Ds should be different than Dt or Ts should be different than Tt. If Ds = Dt and Ts = Tt then traditional machine learning approaches are able to perform well. One dimension along which we could categorize transfer learning approaches is based on the training data that are available for the two domains. Methods that deal with labeled data for both domains are known as *Inductive transfer*. Methods that need labeled data in the source domain and unlabeled in the target domain are known as *transductive transfer learning*. And methods that deal with unlabeled data for both domains are known as *unsupervised transfer* learning. Another difference between these categories is that the *inductive transfer* methods make the assumption that the learning task between the two domains is different and the domains may be related or not. This means that the conditional probability distributions (probability of a class given the training data) between the domains are different or the label spaces are different for example the source domain has binary document classes whereas the target domain has 3 classes to classify the document. On the other hand, methods in *transductive transfer learning* category assume that the learning task is the same for both domains but the domains are different, which means that either the feature space is different, for example different language is used between the two domains, or the marginal distributions are different which means that the source and target domain focus on different topics. Figure 2 shows the assumptions that the two areas do.

3.2 Related work

Sentiment classification is a very domain specific problem. Classifiers trained on one domain do not perform well in another. An example of different domains in this field is reviews from DVD products and reviews from books. The vocabulary will be different or the relation between text and scores will be different, which means that the words in DVD products do not express the same opinions if they appear in book products.

Sentiment classification has focused on transductive transfer learning approaches. The main reason is the assumption that new domains do not have labeled data so transfer learning should perform by taking advantage from the labeled data in the old domain. Approaches can be divided into two main categories: Those that try to customize supervised learning algorithms and those that try to extract *generalisable* features, which are features that are likely to been used by the same way among different domains.

Aue & Gamon, (2005) were among the first who tried to transfer learning among different domains by customizing supervised learning classifiers. Their experiments cannot lead to high performance and indicate that more advanced approaches are needed. The best accuracy was achieved by training an ensemble of classifiers and using a SVM as a meta-learner in order to combine the results. Among others, they tried to train classifiers in the source domain with features that appear in the target domain with no success which reveals that even if we use features from the target domain it does not mean that these features have the same meaning for the source domain.

Two methods that try to extract *generalizable* features from source and target domain proposed by Blitzer et al.(2007) and Tan et al. (2009). Methods in this category are based on two main observations that were nicely described by Raina et al. (2006):

1) The presence of a word w_1 in a document with label Y might make it more likely that words similar to w_1 will occur in other documents with the same label Y.
2) Some rare words are possible to be more or less informative about a document label than common words.

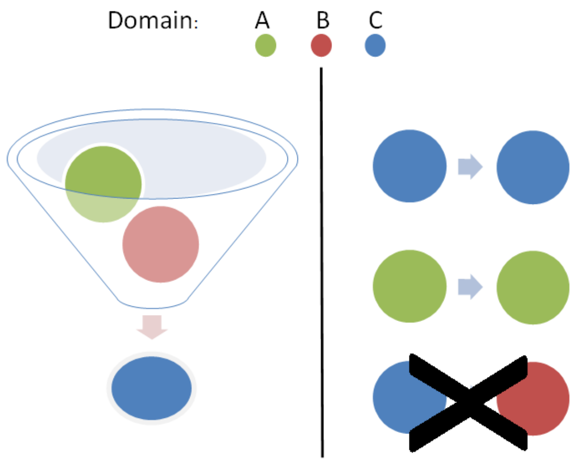
An algorithm which can be trained on a single source domain and adapt knowledge to a target domain proposed by Blitzer, (2007). His algorithm, known as Structural Correspondence Learning with Mutual Information (SCL-MI) uses pivot features. Pivot features are generalized features extracted from the source and target domain. They are selected based on how frequent they are repeated in both domains and also based on the mutual information with the target labels in the source domain. A number of pivot features are extracted and including the initial features from the target domain a new learning problem has to be solved where the aim is to align correlated features in two domains. An extension of this framework includes some labeled instances from the target domain in order to reduce feature misalignments. Results indicate that just a small number of label features can lead to improved accuracy.

Adapted Naïve Bayes is the method proposed by Tan et al. (2009). Similar to the previous approaches this method tries to extract generalized features in order to implement a weighted version of Naïve Bayes classifier. More specifically, firstly features are extracted based on the Frequently co-occurring entropy and then an extension of Expectation Maximization Naïve Bayes algorithm is proposed, which deals with both labeled and unlabeled instances.

3.3 Summary

To sum up, applying traditional machine learning approaches to a different domain results to low performance if the target domain has different distribution from the source domain. Transfer learning area proposes methods that make it possible to overcome this problem. The general concept of those methods, which have been adopted by the Sentiment analysis field, is to extract generalized features and use this information in order to train linear classifiers. If the vocabulary between the source and target domain is not so different, these method will perform well as the generalized features are not domain dependent. On the other hand, the extraction of generalized feature is not easy and depends on the relation between source and target domain.

All of the approaches described in this section do not look on the labels of the target domain, so it is possible to transfer the knowledge to a domain that labeled data are missing. A limitation to this is that they make the assumption that similar or common features have the same probability to be classified in a particular class for both domains. Even in cases that generalized features are extracted it does not mean that this assumption holds. However, methods have been proposed that add some labeled data from the target domain in the source domain, which can lead to improved performance as experiments performed by Blitzer, (2007) indicate.



(a)

(b)

Training and Test data can be from different domains.

Training and Test data must be from the same domain.

Figure 2 Different learning processes between Transfer learning approaches (a) and traditional Machine learning (b) when transfer learning is applied. Figures are adapted from (Pan & Q. Yang 2010)

4 The Proposed Strategy

This section introduces the reader to the various problems that this project aims to solve, the datasets that have been analyzed, the assumptions behind the experimental strategy and the challenges of this project. Added to this, it describes the way I have worked by combining approaches from the bibliography in order to complete the aims and objectives described in section 1.1.

4.1 Datasets and peer review process

I experimented on reviews of research papers submitted to two major conferences: 1612 reviews from the Knowledge Discovery and Data Mining 2009 (KDD) conference and 1247 from the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2012 (ECML). Figure 3 describes the peer review process and Figure 4 presents the details of these datasets.

According to the peer review process, scientists write research papers and they submit them to conferences or journals. Papers are reviewed by referees (in the most of cases three of them) who are experts in the relevant subject. Every review consists of different fields for which the referees need to express their opinion. Examples of such fields are the three strong and three weak points of the paper, the main contributions, detailed comments to the authors etc. Finally, they give a score, which indicates their willingness to accept or reject this paper. The score rating may be different among the conferences. This process is very important in order to ensure that high quality papers will be published. For this project the field which contains the detailed comments to the authors has been used as the review text and the recommended score has been used as the ground truth.

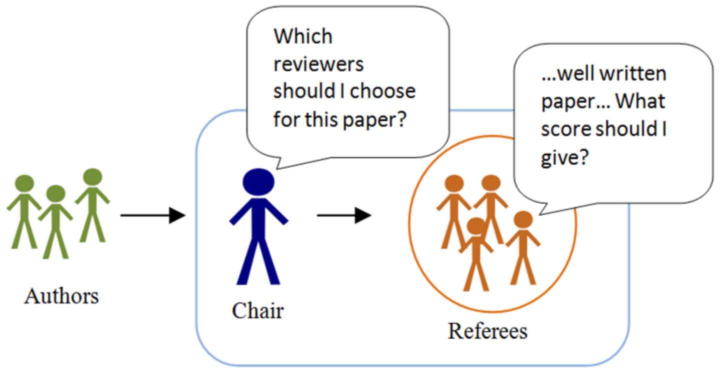

Which reviewers should I choose for this paper?
...well written paper... What score should I give?
Authors
Chair
Referees

Figure 3 Peer review process diagram.

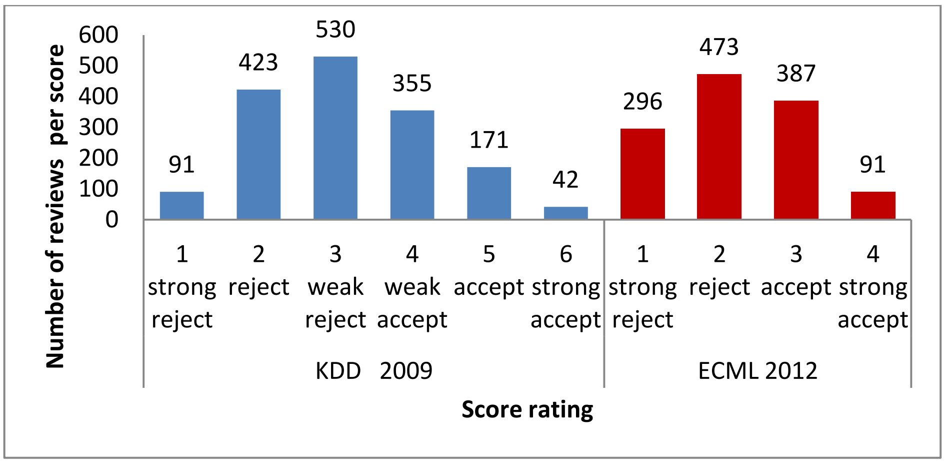


Figure 4 Details of data sets: Conference, Score rating for this conference, number of reviews per score.

4.2 Learning problems applied to the datasets

Three learning problems included their sub problems, as described in section 2.2, have been solved in order to extract opinions, classify, predict numerical scores and rank the reviews submitted to the two conferences. At the same time, all of the problems are applied from KDD to ECML and vice versa, in order to investigate which of them can easier been transferred and compare the way that reviewers express their opinions in the two conferences.While the most of the publications focus every time on only one or two of these problems (we have seen in section 2.4 that the researchers mainly focus on classification and some of them compare classification with regression) for this project I tried to deal with all of them in just a single work.

Reviews of research papers consist of the review text and a review score so the first learning problem is to distinguish reviews that correspond to papers that should be accepted as having positive polarity and reviews that correspond to papers that should been rejected as having negative polarity. The most obvious way to do that is to use classification approaches in order to solve this binary classification problem. The second learning problem is to classify the reviews into three categories, negative neutral and positive. This problem is theoretically more difficult. I continue with the four-class classification problem which is even more difficult. Based on the background knowledge, classification approaches still work but regression approaches have been used as well. As a result, regression, in order to predict review scores is the next intention. Ranking is the final learning problem I am interested to solve. Ranking a set of reviews is enough for this case as it can be used in order to learn the preferences of reviewers (independent of particular scores, as score ratings are different between the two conferences) and investigate if they change among the two conferences. Experiments have been made, which show if the assumptions regarding the difficult and suitability of each problem hold. Furthermore, the most competitive models per task will be presented.

4.3 Challenges

Opinion extraction is a challenging process, which is difficult to be understood both from machines and from humans. This project not only has to deal with the challenges in this field but also with many more as discussed in this section. Challenges can be divided into two general categories: In-domain challenges and out-of domain challenges. In the first case we deal with challenges where learning is applied within the same domain for example, from KDD to KDD, in the latter case we deal with challenge regarding models trained on the first domain and applied to the second domain (from KDD to ECML). Figure 5 summarizes the main challenges regarding this project and the way I handled each of them.

Extracting opinions from plain text can be considered as one of the most difficult natural language processing and text mining problems. The main challenge here is to distinguish facts from opinions as only the latter can give information about the point of view of the writer. Defining the words that give opinions and as a result they are subjective is the main idea that opinion extraction tries to solve. The difficulty of solving opinion mining problems depends on the domain for example Aue & Gamon (2005) maintains that product reviews is an easier domain than user web survey feedback documents. On the other hand, reviews of research papers seem quite a challenging domain. Authors mainly use objective sentences to support their opinions. Even if they want to reject a paper they will underline some strong points in order to encourage the authors to try more. Similarly, for those papers that they give a high score they will refer to some limitations. In order to deal with this challenge I experimented with many different feature extraction methods and also different learning problems (classification, regression, ranking). I chose the state-of–the art learning algorithms and opinion feature extraction approaches as proposed by other researchers.

Even if the features that reveal opinions are extracted, a big challenge is the meaning that the same words have for different authors. Two documents written from two different reviewers even if they have exactly the same words could have a different score. This is known as the author calibration problem (Pang & Lee 2006b). Even the best classifier would fail to predict a correct class. An easier solution would be to train a different model per reviewer in order to investigate if they are consistent with their review text and score unfortunately this was impossible having the specific datasets as every reviewer reviews only 5 to 8 papers which are insufficient for training machine learning algorithms. So, no actions were taken.

Imbalanced class problem is a challenge both to learn within the domain and when we transfer knowledge to a different domain. This problem is due to the fact that some classes have significantly more instances than others. The learning algorithms can be affected by the majority class, which make them unable to learn the minority class (Monard & Batista 2002). In order to deal with this problem I use MNB which takes the prior of the classes into account; in addition to that I searched for evaluation measures that have been recommended as more suitable for imbalanced datasets finally, when I merge different scores in order to solve different problems I try to have as balanced classes as possible.

The two conferences have different score rating. The big challenge is to find those scores that correspond to each other. Different experiments using machine learning algorithms have been performed in order to find similarities and differences. Furthermore, classifiers can be applied to a different domain only if it has the same number of classes with the source domain. There was a need to merge reviews with specific scores in order to have the same number of classes between the two conferences. Merging is performed by considering both the imbalanced class problem and the similarity of the review texts and review scores between the conferences. Finally, I experimented with rankers that are known to perform well independent of the score range.

KDD and ECML are two different conferences. Different authors, different reviewers, ECML is more recent etc. Transferring learning between two different domains is a challenging process. This is why I experimented with many different learning problems in order to investigate which is easier to be applied to a new domain. Finally, a big challenge is that many experiments have been performed; all of them should be evaluated in a common way and be interpreted in order to deliver interesting information regarding the way that authors review research papers. The risk of getting lost in the results was very high and in order to deal with it I developed a common evaluation process and a robust result analysis, which were very important in order to complete this project.

4.4 Summary

This section presented the two conferences that have been used as the training sets during the experimental phase. Moreover, it offered an introduction to the problems that will be solved and remarked the main challenges and the way they have been handled. The next section describes in details the processes and the algorithms that have been chosen, before the results of this research are presented.

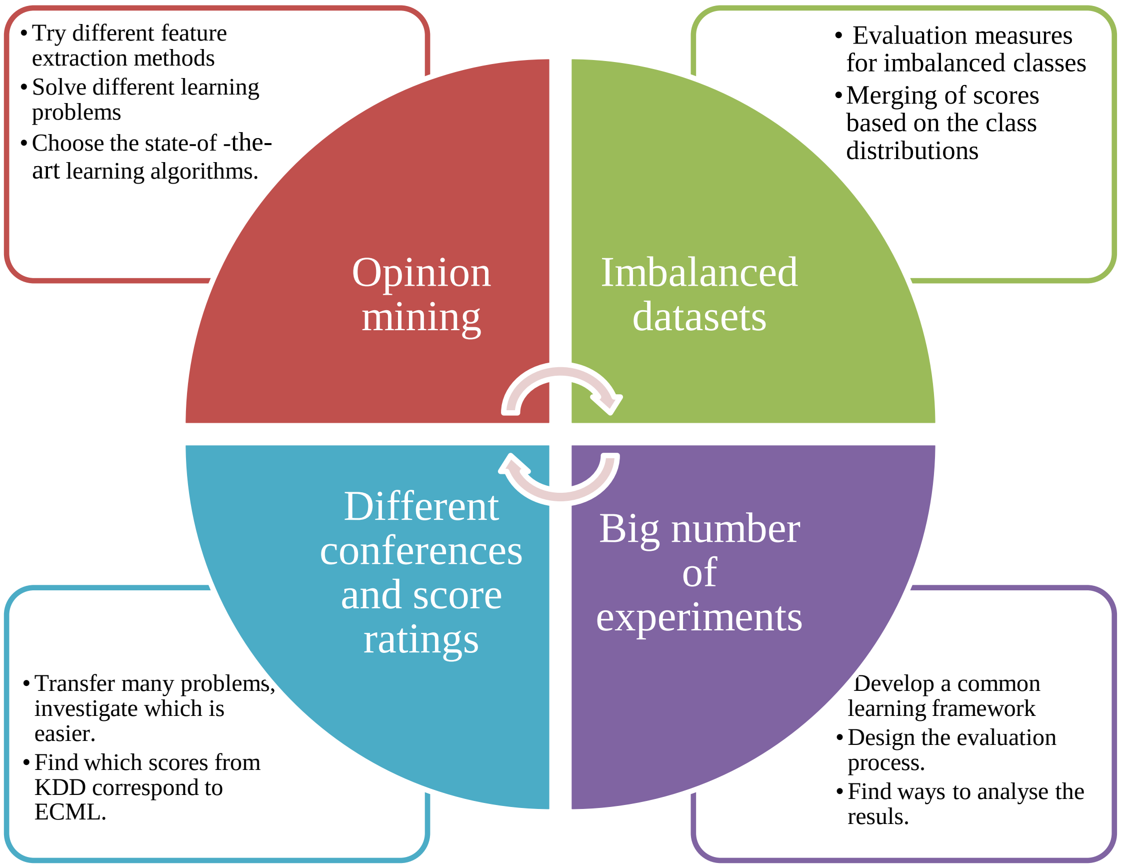

• Try different feature extraction methods
• Solve different learning problems
• Choose the state-of -the-art learning algorithms.
• Evaluation measures for imbalanced classes
• Merging of scores based on the class distributions
Opinion mining
Imbalanced datasets
Different conferences and score ratings
Big number of experiments
• Transfer many problems, investigate which is easier.
• Find which scores from KDD correspond to ECML.
Develop a common learning framework
• Design the evaluation process.
• Find ways to analyse the resuls.

Figure 5 Main challenges regarding the project and steps that have been taken in order to handle them.

5 Experimental Setup

This section provides an overview of the software and machine learning algorithms that have been chosen in order to solve the different learning problems and complete the scope of this project. Furthermore, it explains the complete evaluation process and the statistical tests performed during the experiment phase.

5.1 Feature extraction

Machine learning algorithms are able to give good results as far as the feature space is good. In section 2.3 we saw that feature extraction is a very important part of the whole process in order to create datasets that contain opinionated features. I experimented with various feature extraction methods as described in this section. There are not published results regarding the specific datasets so there is no evidence which of them is the most suitable. For this reason, I started with very simple approaches like unigrams and bigrams and based on the performance I continued with the choice of alternative ones. Weka (Witten & Frank 2005), Stanford Natural Language Processing toolkits, the General Inquirer (Stone et al. 1966) and SentiWordNet lexicon (Esuli & Sebastiani 2006) are the libraries and lexical resources used during the preparation phase, for feature extraction and conversion to feature vector representation as described in Figure 1.

The Stanford Tokenizer is initially used in order to split the reviews from the two conferences into sentences and normalize each of them. Normalization includes token transformations like parentheses becoming -LRB-, -RRB-, ascii quotes, latex quotes transformations and many more. This process is important in order to obtain a better form of the initial reviews that can better be divided later into words or ran through different parsers. The Stanford Tokenizer is a part of other software and is not separately distributed but it can be found together with other distributions like the Stanford Part of Speech tagger (Toutanova & C. D. Manning 2000).

Based on the findings of other researchers, specific parts of speech like adjectives and adverbs reveal opinions, for this reason and similarly to the work of Pang et al. (2002) I use Stanford Part of Speech tagger, in order to tag every word with its parts of speech (Toutanova & C. D. Manning 2000). In addition to that, I experimented with approaches that focus on the syntactic analysis of a sentence, which is provided by parsing the text using Stanford typed dependency parser (Marneffe et al. 2006). The parser returns pairs of words that reveal typed dependencies based on grammatical relations like *subject* or *indirect object*. This is a more clever way compared to simply extracting pairs or triples of sequential words. Finally, I experimented with features extracted by opinion lexicons. The General Inquirer lexicon[3] is used in order to replace every word that is detected to be an overstatement or understatement with a single tag: OVS for overstatement and UND for understatement. Finally, I obtain some first statistical results regarding the polarity of each review by using the SentiWordNet lexicon (Section 6.2.1).

The final step is to convert the reviews into a suitable format for the learning scheme. The StringToWordVector filter provided by the Weka machine learning library converts input texts to vectors of unigrams or n-grams and gives the final feature vector representation. More specifically, it offers a choice of the way that the vectors will be represented (frequency counts, tf-itf) and also performs some simple preprocessing like the exclusion of very frequent words or stop words. APPENDIX C summarizes all of the feature sets that have been extracted regarding this project and some more annotations.

[3] GI is a dictionary that annotates English word senses. It includes tags that label them as positive, negative, overstatement understatement etc.

5.2 Machine learning algorithms and parameters

I experimented with different learning schemes suitable for the different learning tasks. I chose those algorithms that have been preferred by other researchers in this field and also have been recommended as presenting the best performance, as discussed in section 2.4. Figure 6 shows all of the learning algorithms that have been used and APPENDIX A gives more details regarding each of them. The WEKA machine learning library was used as the source of learning algorithms for classification and regression and the RankLib library was used as the source of RankNet. This section describes the learning algorithms, the different parameters used to train each of them and the process of choosing the corresponding parameters.

- **Classification**. Two machine learning algorithms Multinomial Naïve Bayes (MNB) and Sequential Minimal Optimization Algorithm (SMO) are used in order to build classifiers. MNB is used with default values for all of the parameters. SMO is WEKA implementation of Support Vector Machines (SVM). SVMs can perform better if their parameters are optimized, for this reason I tested different combinations of the parameters separately for every learning problem. More specifically, normalization is always turned on as recommended by Graf & Borer (2001). I tested 8 values for the complexity parameter {10−5, 10−4, 10−3, 10−2, 0.1, 1, 10, 100}, and 10 different kernels: linear kernels, 1 polynomial kernel (of degree 2) and 8 radial kernels (gamma ∈ {0.001, 0.005,0.01, 0.05, 0.1, 0.5, 1, 2}). The pair of complexity parameter and kernel that presents the highest accuracy in the 10-fold cross validation is selected and used to classify all of the datasets in the given learning problem. One could argue that this is unfair, because I tune a parameter by peeking at the test sets. However, I do not use the best combination for each feature set, rather a fixed pair of values, which could be suboptimal for some of them. In a future work a better approach could deal with it as a part of the training process for example full grid search could be applied over many combinations of parameter pairs as recommended by Hsu et al. (2003), where for each of the 10 train/test splits, a validation fold can be set in order to gather the optimized parameters. Finally, One Versus One technique is used for multiclass classification, which is the default that Weka uses.
- **Regression**. Three different algorithms are used in order to build regression models SMOreg which implements SVM for regression problems, Additive Regression(AR) and Bagging (Bag).The values of the parameters for the SMOreg were chosen with the same way as described in the previous paragraph. To the best of my knowledge Bag and AR are two ensemble methods, which have never been used by the opinion mining field. *Ensemble methods* (Dietterich 2000) combine many models, known as experts, in order to take the final decision. They have been a very popular research topic during the last decade due to the fact that they offer an appealing solution to several interesting learning problems such as improving prediction accuracy, scaling inductive algorithms to large databases and learning from concept-drifting data streams (H. Wang et al. 2003). Model trees were used in order to construct the homogeneous ensemble as recommended by (Witten & Frank 2005: p. 325-327). All of the parameters are used with default values.
- **Ranking**. MNB, SMOreg and AR have also been used as rankers. For MNB I convert the estimated probabilities into scores as discussed in section 2.4.3. I compared these models against the RankNet (RN) algorithm (also discussed in section 2.4.3). Neural networks are robust approaches but they require many hours in order to be trained. Due to limitation of time and in order to reduce the computational complexity I reduced the number of trained epochs to 10 and I let 10 nodes per layer instead of the default values, which were 40 and 20 respectively. For each of the 10 train/test splits, the training set is further split in order to have a validation fold the final model is the one that performs best on the validation data.

5.3 Evaluation methodology and baselines

Evaluation of the different methods is the most important aspect of solving machine learning problems. The evaluation of the performance is needed for mainly two reasons: Comparisons with other learning models and optimization of a specific model. In the first case, we want to compare the results between different models in order to see which of two performs better and in what points. In the second case, we want to observe the limitations and the strong points of a specific learning scheme in order to optimize some parameters or realize why something goes wrong. Finally, we want to evaluate methods in order to understand better the data and the difficulties of a problem. The whole process of learning and transferring different conferences was quiet complicated. I will explain the general process and the principles that have been used in order to evaluate the performance. Moreover, I will emphasize to some limitations and propose some alternative possibilities. After that, the section is further divided into two subsections: Evaluation Measure subsection describes the measures that have be chosen to evaluate models depending on the learning problem and statistical testing subsection describes all of the statistical tests performed regarding this project.

Many researchers underline the importance of evaluation of machine learning algorithms in order to make “real progress” in Machine Learning field and the fact that some researchers do not take into consideration how significant this process is (Witten & Frank 2005: p. 143) (Japkowicz & Shah 2011: p. 2-4). Taking everything into account, sufficient time has been spent in order to develop a common learning framework for the evaluation of the multiple schemes according to suggestions found in bibliography. A helpful source of information has been found in Flach(n.d.) and Demsar (2006). Flach’s chapter analytically describes the whole process of evaluation and suggests appropriate solutions regarding the learning problem while Demsar’s paper focuses on statistical testing. The choice of learning algorithms and datasets has been discussed earlier in this section. For the rest, we focus on the other three components.

Figure 6 presents the whole learning and evaluation process and APPENDIX H contains a part of the Java code that corresponds to this process. Conference 1 and Conference 2 correspond to the reviews submitted to KDD and ECML respectively. Reviews from the first conference are used in order to build a model. This model is tested on reviews from the same conference I will refer to this as the in-domain evaluation, which is the common way that traditional machine learning approaches are evaluated. The same model is applied to the second conference and its performance is also evaluated there. This is the out-of domain evaluation or the transfer learning phase, where machine learning approaches are expected to fail. More specifically, reviews from the first conference are converted into feature vectors. A machine learning algorithm takes as input the feature vectors and builds a model. The algorithm is chosen based on the learning problem we want to solve. The model is used for in domain and out-of domain evaluation. Evaluation measures have again been chosen depending on the learning problem. The same process is repeated but now ECML is used as the source domain and KDD is used in order to apply transfer learning.

Stratified 10-fold cross validation is used for the in-domain evaluation. For the out-of domain evaluation we use the whole target dataset without splitting it. More specifically, during the in-domain evaluation 10 different models have been built, one for every repetition for the 10-fold cross validation. These 10 models are separately used to classify, rank or predict numerical score on the whole dataset of the second domain. The final performance is the average over the 10 repetitions.

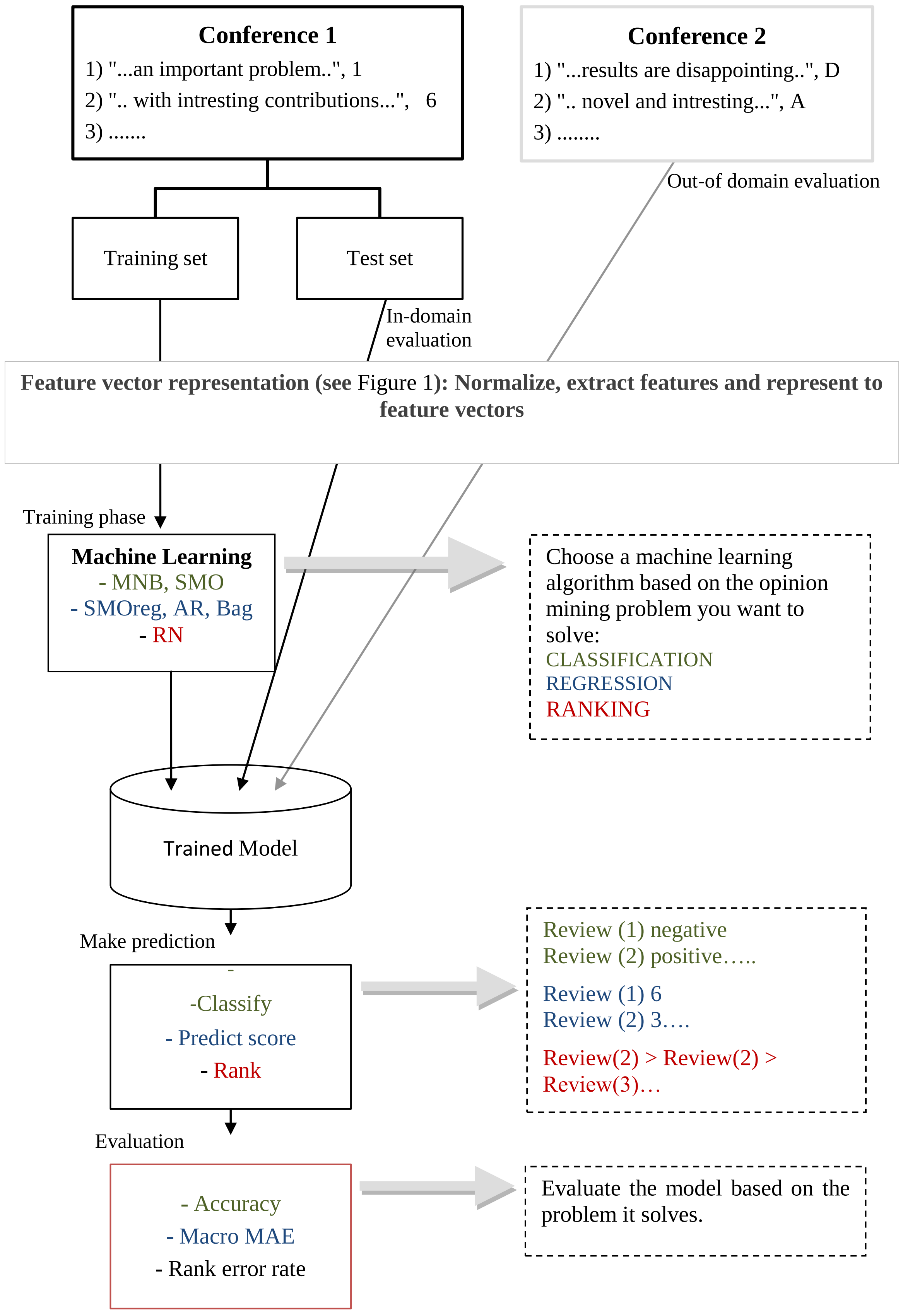

Conference 1
1) "...an important problem..", 1
2) ".. with intresting contributions...", 6
3) ....
Conference 2
1) "...results are disappointing..", D
2) ".. novel and intresting...", A
3) ....
Out-of domain evaluation
Training set
Test set
In-domain evaluation
Feature vector representation (see Figure 1): Normalize, extract features and represent to feature vectors
Training phase
Machine Learning
- MNB, SMO
- SMOreg, AR, Bag
- RN
Choose a machine learning algorithm based on the opinion mining problem you want to solve:
CLASSIFICATION
REGRESSION
RANKING
Trained Model
Make prediction
-Classify
- Predict score
- Rank
Review (1) negative
Review (2) positive…..
Review (1) 6
Review (2) 3….
Review(2) > Review(2) > Review(3)…
Evaluation
- Accuracy
- Macro MAE
- Rank error rate
Evaluate the model based on the problem it solves.

Figure 6 The learning framework

The decision regarding the in-domain evaluation was an easy task as much of bibliography is available regarding the appropriate way of performing machine learning experiments and evaluating them when the source and target domain come from the same distribution. Cross validation is the method proposed by many researchers and can give meaningful results if the training and test data are sampled according to the constructions (Flach n.d.). Stratified cross validation ensures the same proportion of class labels in each fold and prevents from taking biased results. Feature extraction and feature selection is based only on the training data, so at no point are test data seen by any of the approaches. The only exception is the SMO parameters optimization as discussed in the previous section.

The decision regarding the out-of domain evaluation was a more difficult task as there is not much discussion on this area. Apart from the way that I have decided to do the evaluation, two alternative solutions are also possible. We could train a model on the whole dataset of the source domain and evaluate it on the target out – of domain dataset instead of evaluating 10 different models. However, this could lead to overestimated results that could not give a fair comparison between the performance of the same model when it is evaluated on different domains. The second alternative could be to split the second domain into 10 non-overlap folds and test each model built on the source domain to the corresponding fold. These 10 non-overlap folds should be the same with those used for in-domain evaluation when the second domain is used as the source domain. This alternative gives the opportunity to make pair comparisons per fold in order to compare if the transfer learning performance is equivalent with the performance when the model is both built and tested on its own native domain. It should be noted that when I evaluate the out-of domain performance no splitting is performed so we cannot do pair comparisons. The overall results should not really be different and we cannot assume that the one approach should be preferred compared to the other. The only thing is to be careful with the way we perform statistical tests for this reason the decisions that have been taken are described later in this section.

5.3.1 Evaluation measures and analysis

After deciding about the principle for performance evaluation, a quality or evaluation measure has to be chosen which can quantify the success of a learning scheme. In this section I describe the different measures that have been preferred based on the "*experimental objective*" and the learning task as recommended by (Flach n.d.). More specifically, Flach distinguishes the evaluation measures from the experimental objective and underlines that evaluation measures should be chosen based on the *experimental objective*.

In classification the *experimental objective* is to find those models that classify correct the most of the reviews. Accuracy, which counts the correct classifications, is used as the evaluation measure. However, knowing only the number of mismatches on a number of instances is not sufficient. Among others our aim is to explain why wrong classification happens and which classes can easier be predicted as a result, a process that reveals different kind of errors is essential. Confusion matrix is a common way used to illustrate these mismatches (Table 2). Based on the values that a n*n confusion matrix produces one of the metrics that can be extracted is recall, which can be used in order to annotate the performance of a model (Mingzhi et al. 2009). According to Tang et al. (2009) in practice the most important evaluation measure, in sentiment analysis, is accuracy and many applications prefer to sacrifice recall for accuracy as it is more important to have an equal number of positive and negative classification instead of trying to classify positive reviews better than negative. For this project the aim is to optimize accuracy but weighted recall and confusion matrix are calculated in order to explain the performance of every model. Weighted recall is the recall multiplied by the prior of this class.

Table 2 Confusion matrix and weighted recall.

a	b	c	←Classified as	
(na)	*	*	**a**	**Na**
*	(nb)	*	**b**	**Nb**
*	*	(nc)	**c**	**Nc**
Total # of instances: Na+Nb+Nc = **N**				

$$Weighted\ Recall\ i = \frac{ni}{Ni} * \frac{Ni}{N}, i = a, b, c$$

ni: The # of instances that belong to class i and correctly have been classified to this class.
Ni: The total # of instances belonging to class i.
N: The total # of instances.

In regression the *experimental objective* is to find those models that predict numerical scores, whose predictions are as close as possible to the ground truth. There exist many measures commonly used in machine learning field like mean squared error, absolute error, relative squared error etc. For this project the Mean Absolute Error (MAE) is used as the measure to be optimized. More specifically, I preferred the macro averaged version of MAE as recommended by Baccianella et al. (2009a) to be more suitable for imbalanced datasets.

The final learning problem is ranking. The *experimental objective* here is to find those models that make the least rank errors. The rate of the rank error is estimated based on *the utility-based rank loss* as proposed by Fürnkranz & Hullermeier (2011: p.227 - 229):

$$\mathbf{loss} = \begin{cases} 1, if\ (y_1 - y_2) * \left(f(\overrightarrow{x_1}) - f(\overrightarrow{x_2})\right) < 0\ \ (a) \\ 0.5\ , if\ (y_1 - y_2) * \left(f(\overrightarrow{x_1}) - f(\overrightarrow{x_2})\right) = 0,\ y_1 \neq y_2\ (b) \\ 0, otherwise \end{cases} \quad \mathbf{error} = \frac{a + b}{total\ \#\ of\ pairs}$$

This loss function gives different penalties which are proportional to the importance of the error. For example, bigger penalty will be given if the model gives a score $f(\overrightarrow{x_1}) > f(\overrightarrow{x_2})$ when the true labels are $y_1 < y_2$ and smaller penalty if the model predicts that $f(\overrightarrow{x_1}) = f(\overrightarrow{x_2})$. y_1, y_2 are the ground truth review scores and $f(\overrightarrow{x_1}), f(\overrightarrow{x_2})$ are the scores that the ranker assigns to the instances $\overrightarrow{x_1}, \overrightarrow{x_2}$.

A different *experimental objective* is to investigate if regression models perform better as rankers. Pearson's correlation is the measure that counts the linear relation between two variables. Spearman's correlation is the non-parametric alternative that uses the ranks instead of the exact labels. Due to limitation of time I did not make this comparison however it could be considered in the future work. Table 3 summarizes the evaluation measures that have been optimized per learning problem.

Due to the fact that many models are built and need to be compared I have developed a systematic way to analyze the results in order to be easily understood by the reader. This process will be described during the experimental section.

Table 3 Evaluation measures per learning problem.

LEARNING PROBLEM	CLASSIFICATION	REGRESSION	RANKING
Experimental objective	1) Correct classification 2) Model's failure	1) Correct prediction 2) Comparison with ranking	1) Rank error 2) Comparison with regression
Evaluation measure	1) Accuracy 2) Weighted recall	1) macroMAE 2) Pearson's correlation	1) rate of Rank Error 2) Spearman's correlation

5.3.2 Statistical testing

Evaluation measures are able to show a good view of the performance of a learning scheme. However, statistical tests are needed in order to reveal if this performance is significant and able to predict similar results in future data. Added to this, statistical tests are necessary in order to test if the difference in performance among different approaches is statistically significant. T-test and Wilcoxon signed ranks test are two statistical tests commonly used in machine learning field. Depending on the comparisons the appropriate tests have been chosen as described in the rest of this section. Figure 7 summarizes all of the statistical tests performed regarding the needs of this research.

Two tailed paired t-test: Paired t-test can be used in order to compare different models trained on the same dataset (Demsar 2006). In this project paired t-test is applied in order to compare the performance within the same domain. More specifically, during the experimental section we search for those models and feature extraction approaches that lead to higher performance. For example, we are interested in answering whether training MNB using unigrams or MNB using bigrams leads to better classification when source and target domain come from the same conference. As discussed earlier, I use 10-fold cross validation for the in-domain evaluation as a result comparing pairs of folds between the two different models can show if the models are equivalent or not. The results obtained by 10-fold cross validation regarding the evaluation of the two learning schemes can be treated as two random variables. The paired t-test makes two assumptions about the data:

1. The differences between the errors of the two random variables are normally distributed.
2. The values of each random variable are independent.

Within cross validation folds we can assume that the errors are normally distributed. On the other hand, different folds do not really contain completely independent data because the data tested are non-overlapped but in order to train a model the training sets within the different folds overlap. However, researchers confirm that the test can give strong results even if this constraint is not completely satisfied (Demsar 2006).

Unpaired t-test (unequal variance): Unpaired t-test is useful in order to compare different models trained on different datasets (Demsar 2006). This test is used in order to compare in-domain results with the corresponding out-of domain results. For example, a model built with reviews submitted to KDD and tested on KDD need to be compared with the corresponding model which was built with reviews from ECML and tested on KDD. This comparison will show if the two models perform the same and as a result if transfer learning is possible. Paired t-test cannot be used here because for the in-domain evaluation the dataset is split into 10 non-overlap folds while for the out-of domain evaluation the whole dataset is tested within every fold. For both cases we have 10 results each one corresponds to a different fold of 10 fold cross validation. However, they are not comparable because for the in-domain model each fold contains the 1/10 of the whole dataset of the target domain, but for the case that we apply transfer learning each fold contains the whole dataset of the target domain. Similarly to the paired t-test this test makes two assumptions:

1. The two random variables are normally distributed.
2. The two populations to being compared are independent.

The sampling process when we apply transfer learning is not completely correct because the same datasets are used which means that the folds are not independent. On the other hand, each fold is evaluated with a different model which results to independent measures even if the data are not independent. Within cross validation folds we can assume that the errors are normally distributed. One could argue that the second assumption is not satisfied because the in-domain and out-of domain datasets are not independent however; they are evaluated by independent models[4] so the measures are independent and the test is applicable.

Wilcoxon signed rank test: Wilcoxon signed rank test compares two algorithms over multiple datasets (Demsar 2006). One example that this test has been used is to compare SVM versus MNB over different feature extraction approaches. Over multiple datasets we cannot assume that the errors are normally distributed, so paired t-test is not safe here. Wilcoxon signed rank test is the non-parametric alternative of paired t-test which leads to more robust results if its normality assumptions are violated (Demsar 2006).

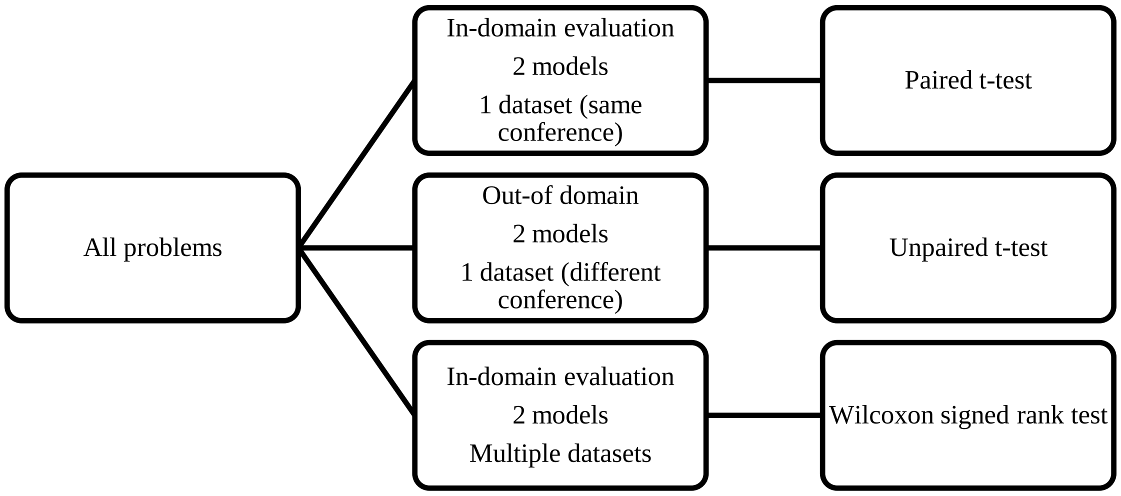


Figure 7 Summarization of the statistical tests used in experimental section.

Finally, as baselines for classification problems I use the majority class and for regression the mean score. Accuracy is an evaluation measure that can be misleading if the priors of the classes are different as the learning scheme can present high accuracy just because it has learned to classify everything as belonging to the majority class (Monard & Batista 2002). For ranking the baseline is equal to 1, which is the case that all of the pairs have wrongly been ranked.

5.4 Summary

In this section I described the way that different feature sets have been extracted, the learning algorithms and the parameters that have been used in order to build models and finally, the evaluation measures and the statistical tests that have been selected in order to evaluate the outcome of this research. Figure 6 presents the way that all of these have been combined in order to create the learning framework that has been used for experiments regarding the scope of this project. Finally, APPENDIX B summarizes the list of software adopted for this project.

[4] Models trained on KDD and applied to ECML are independent of models trained on ECML and tested on ECML.

6 Experiments and Evaluation

The whole experiment process has been divided into four parts. Firstly (Section 6.1), I present some preprocessing techniques that have been applied and the initial experiments that led me to decide about the feature vector construction. Secondly (Section 6.2), I take a closer look on the dataset, in order to obtain some first statistical results regarding the two conferences. Furthermore, I investigate how difficult will be to solve the three learning problems. The third part (Section 6.3) is the most extensive, where the learning framework is instantiated with different combinations of feature sets and machine learning algorithms in order to learn and transfer the problems. Last but not least, Section 6.4 shows a different approach of transferring models that failed to be transferred during the binary classification task.

At this point I would like to sum up some notations and terminology. In-domain evaluation is referred to models that have been trained and evaluated on reviews from a specific conference. As a result, there are two possible cases: models trained and evaluated on ECML (ECMLtoECML) and models trained and evaluated on KDD (KDDtoKDD). Out-of domain evaluation is referred to the evaluation of models that have been trained on reviews from one conference and tested on the reviews of a different conference. The two possible cases are ECMLtoKDD and KDDtoECML.

Binary classification, multi-class classification, regression and ranking are the problems that will be learned and transferred as discussed in Sections 2.2 and 4.2. Table 4 summarizes different ways that the scores have been treated, which results to different learning problems. KDD contains reviews with a score range from 1 to 6 (1 = strong reject, 6 = strong accept) while ECML contains reviews with a score range from 1 to 4 (1 = strong reject, 4 = strong accept). In order to avoid confusion, I will refer to the scores of ECML with the letters D to A (D = strong reject, A = strong accept). The second column shows which scores have been merged for KDD, while the third column shows the corresponding scores for ECML. The fourth column shows the nominal label that has been assigned to the scores with values as shown in columns two and three, while the last column shows the continuous value that corresponds to the nominal value. Nominal labels are used in order to treat the problem as classification, while continuous labels are used in order to treat the problem as regression or ranking.

For example, in order to build models regarding the three-class classification problem, for KDD reviews with scores 1,2 are treated as *negative*, reviews with scores 3,4 are treated as *neutral* and reviews with scores 5,6 are treated as *positive*. Similarly, for ECML reviews with score D are treated as *negative*, reviews with scores B,C are treated as *neutral* and reviews with score A are treated as *positive*. In a similar manner, to build regression models that predict numerical scores on a three scale range. For KDD, reviews with scores 1,2 are treated as having *score 1* reviews with scores 3,4 are treated as having score *2* and reviews with scores 5,6 are treated as having *score 3*. Lower scores correspond to negative overall opinion and higher scores correspond to positive overall opinion.

It should be remarked that the way the scores have been merged is not straight forward. The only exception is the merging regarding the binary classification, where the separation is based on papers that have been accepted or rejected. The two conferences have different score ratings as a result, I had to investigate which scores of KDD correspond to scores of ECML based on the review text. In section 6.2.2 some early experiments, using MNB, show the similarity between reviews with specific scores. Added to this, I tried to merge scores in order to have the same number of instances to the corresponding classes between the two conferences.

Table 4 This table shows how the reviewing scores have been merged in order to solve different learning problems. KDD contains reviews with a score range from 1 to 6 while, ECML contains reviews with a score range from 1 to 4. In order to avoid confusion I will refer to the scores in ECML with the letters D to A.

	Reviews with scores			**are treated as**	
Learning problem	(KDD)	(ECML)		(Nominal class label)	(Continuous class label)
Binary classification	1,2,3	D,C	⟶	negative	-
	4,5,6	B,A	⟶	positive	-
Three class classification Three-score regression/ ranking	1,2	D	⟶	negative	1
	3,4	C,B	⟶	neutral	2
	5,6	A	⟶	positive	3
Four class classification Four-score regression/ ranking	1,2	D	⟶	Strong negative	1
	3	C	⟶	negative	2
	4	B	⟶	Weak positive	3
	5,6	A	⟶	positive	4
Full-score regression/ ranking	1 - 6	D - A	⟶	-	The real scores

6.1 Feature vector construction

The first step of the experiment phase was to convert every review to its feature vector representation, which is recognizable and meaningful by the machine learning schemes. As discussed in previous sections every review is ran through different parsers in order to normalize it or to tag it with useful information. In this section, I describe some preprocessing techniques that have been applied to the reviews and the way that they are finally represented by feature vectors. The whole process has been divided into three different parts as follows:

6.1.1 Basic preprocess

Before the extraction of features some basic preprocessing techniques have been repeated in order reduce the feature space without losing information. Every review is down cased and after that I perform stemming, remove punctuation except for question mark, exclamation mark, full stop, brackets, dollars, - and #, numbers are treated as separate lexical items, so every number is replaced by the tag: TAGNUMBER. I do not remove any stopwords because including them slightly increases the performance, which shows that the distribution or use of them is different between positive and negative reviews. I assume that infrequent features are paper dependent and I remove those that occur less than k times per class/score. The value of k has been chosen separately for different learning tasks and feature sets (unigrams, bigrams, POS, Stanford dependencies), based on the results on 10-fold cross validation using reviews from KDD. The same k values have also been used for ECML. Table 5 shows the value of k for every learning problem and feature set. The cardinality of vocabulary corresponds to KDD.

It should be noted that the initial vocabulary for the 2 class problem including all unigrams is equal to 15046. This number will be huge if we try to extract bigrams. However, according to

Table 5, if we remove words that occur less than two times per class then the vocabulary reduces to 4235 words. These simple methods lead to a vocabulary size which is reasonable in size and can simplify the calculations. At the same time, the big reduction of the vocabulary size shows that many words used by the reviewers are paper dependent, which is something expected as reviewers of research papers do not only say if they reject or accept the paper but they also comment on specific points of the paper. Finally, it should be noted that all of the reviews have initially ran through different parsers as discussed in section 5.1 and after that preprocessing techniques have been applied as the parsers are sensitive to some techniques like stemming.

Table 5 The number of infrequent words removed per learning problem and per feature set.

Learning problem	**unigrams or unigrams with POS tags**	**bigrams or bigrams with POS tags**	**Stanford dependencies**
k value for 2 class	2	4	2
Vocabulary size	4235	8850	21248
k value for 3 classes /scores	2	2	2
Vocabulary size	4097	23554	19417
k value for 4 classes /scores	2	2	2
Vocabulary size	4129	24081	20064

6.1.2 Feature vector representation

In section 2.3 I referred different ways of representing the data space. The conclusion was that the most suitable depends both on the machine learning algorithm and on the dataset. The data representation, chosen for this project, is the term frequency. According to the formula below, each feature is represented by a weight based on its relative frequency in the document, normalized by the length of the document.

$$term\ frequency = \frac{(1+\log(n_{ij}))}{(1+\log(n_j))}, \quad \begin{aligned} n_{ij} &= \text{ number of occurrences of term i in document j} \\ n_j &= \text{ length of document j} \end{aligned}$$

Frequency term representation was chosen after experiments among different feature vector representations: Term presence, Term Frequency times Inverse Document Frequency (TF-IDF) and also using logarithms and without logarithms. Reviews submitted to KDD have been tested. Those with scores 1,2,3 were treated as negative and those with scores 4,5,6 were treated as positive. SMO and MNB have been trained using unigrams. Table 6 shows the accuracy percentage of the two models for the five different representation approaches. The last row shows the average rank of the performance and I analyze the results in terms of this rank. Term frequency presents the best results both in terms of average rank and in terms of the highest accuracy for both MNB and SMO. On the other hand, TF-IDF results to the lowest accuracy, while term presence performs well for MNB but leads to the worst results for SMO. Note here that this outcome cannot be considered as significant or stable because different feature representation approaches have been tested based only on unigrams extracted from reviews submitted to KDD. It is possible that different feature sets or reviews from ECML would be better described by different representations. However, for this work I adopt this strategy different ways could be investigated in the future.

A final observation here is the low performance of SMO, which is opposite to what one would expect based on the findings of other researchers. The problem is that default parameters have been used which are suboptimal. Parameter tuning is really essential for SVM and it has been taken into consideration during the next experiments.

Table 6 Accuracy percentage and average rank for different feature vector representations.

	Frequency	Frequency(log)	Presence	TF-IDF	TF-IDF(log)
MNB	63.83%	**65.94%**	65.32%	61.91%	61.60%
SMO	61.10%	**61.35%**	59.74%	60.11%	60.92%
Rank	2.5	1.0	3.5	4.0	4.0

6.1.3 Feature selection

All of the simple preprocessing methods discussed earlier lead to a vocabulary size which is reasonable in size and can simplify the calculations. In this section, I further investigate if feature space reduction can lead to a smaller more meaningful feature set. I experimented with MNB and SMO but this time I calculated the accuracy both for unigrams and for bigrams. Reviews submitted to KDD are used as the training set. The models are evaluated both on KDD and on ECML.

In the previous section, I underlined the importance of the parameter optimization for SMO as a result; I search for the best combination of kernel and complexity parameter (as described in section 5.2). More specifically, instantiating SMO with RBF kernels and high values for the complexity parameter lead to the best overall performance. On the other hand, SMO with linear kernels and complexity parameter equal to 0.1 led to a little lower performance. The fact that high values for the complexity parameter can lead to over fitting made me to prefer the latter parameters (Hsu et al. 2003).

I select the k-highest ranking features based on their information gain in order to reduce the feature space. The two figures below show the accuracy percentage regarding the in-domain and out-of domain evaluation for different values of k, when the algorithms are trained using unigrams and bigrams respectively. Default is referred to the case that no feature reduction has been performed. I start the analysis from the first figure, where the algorithms have been trained using unigrams. In-domain evaluation of MNB shows that as far as the number of k reduces the performance slightly increases and after that decreases. There is a peak of 67.00% for k equal to 500 as opposed to the default value of 65.94%. Similar performance presents SMO but the best overall performance results for k equal to 2000 (67.00%) as opposed to the default accuracy of 65.63%. Regarding the out of domain evaluation, MNB shows slightly better performance as far as the number of features decreases while, SMO shows some fluctuations for higher values of k but finally the accuracy is decreased when the k takes lower values.

The second graph shows the performance of the two algorithms when they are trained using bigrams. Regarding the in-domain evaluation, both algorithms present the best overall performance if no feature selection is performed. Many researchers underline that higher order n-grams should be used in a careful way and always in combination with unigrams because data sparseness is a major problem when they are used on their own (V. Ng et al. 2006, Baccianella et al. 2009b). It is possible that the low performance when the features are reduced is due to this problem.

The final decision, based on these results, was to use the whole feature set without performing any feature selection. More specifically, feature reduction sometimes improves the accuracy and sometimes reduces it. The number of k depends on the learning algorithm and on the feature set. Furthermore, feature space reduction is mainly performed to decrease the time complexity. There were not time performance issues as the feature space was quiet small and the algorithms could be trained and tested very fast. Finally, SMO is a classifier known to be robust with high dimensions. For these reasons, machine learning algorithms were left to take care of the whole feature space.

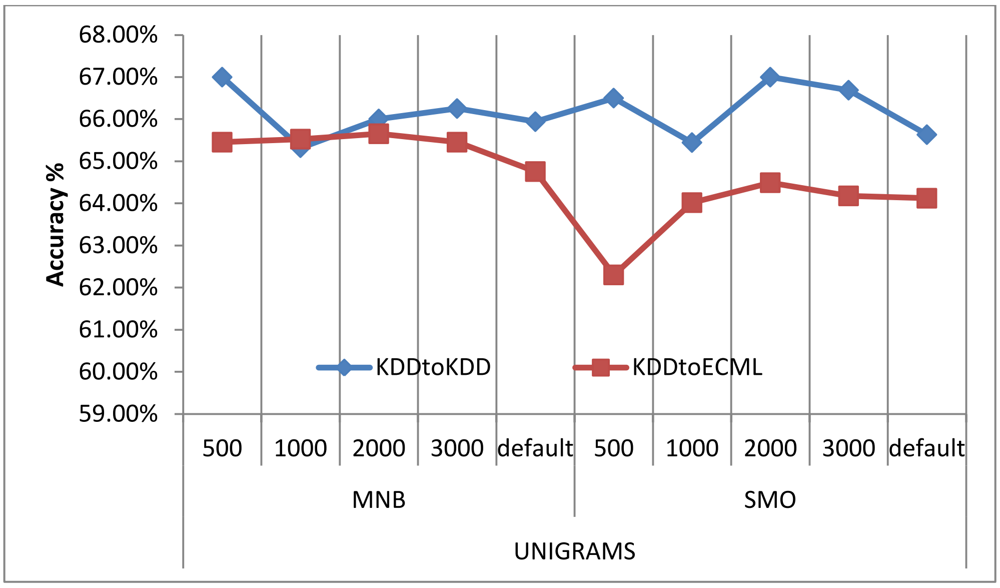


Figure 8 Accuracy percentage for different cutoffs of unigrams with the highest information gain.

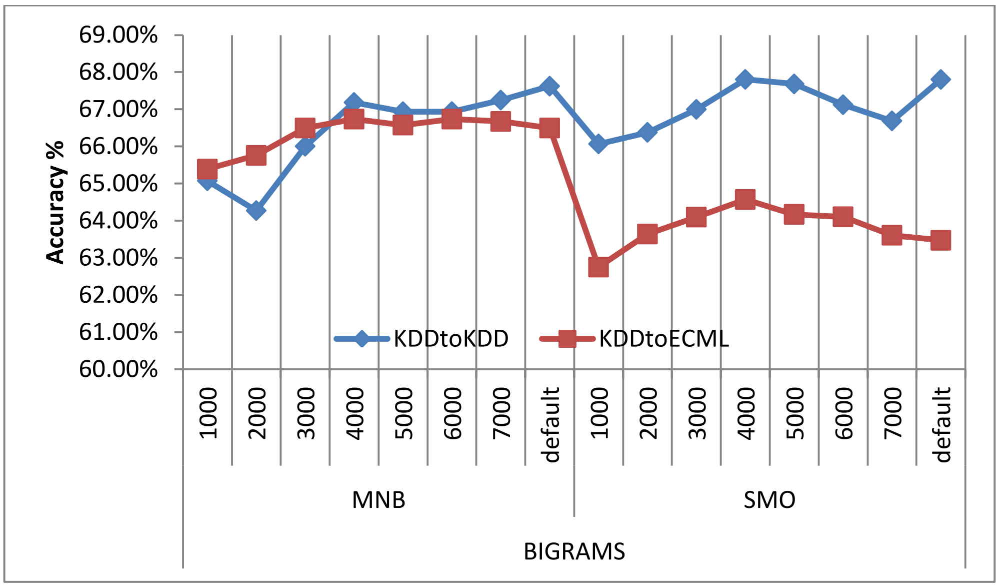


Figure 9 Accuracy percentage of different cutoffs of bigrams with the highest information gain.

6.2 Corpus annotation

The previous section, described all of the decisions that have be taken in order to preprocess the reviews and convert them to feature vector representation. This is the last section before we pass to the main part of this project which deals with solving and transferring the three learning problems. This section shows some first statistical results for the two conferences and solves some theoretically simpler problems in order to give a better understanding of how difficult opinion extraction from reviews of research papers is. Last but not least, similar to Kim et al. (2006) different classes of features are analyzed that are most possible to capture opinions like *structural* (number tags, review length), *lexical* (n-grams), *syntactic* (Pos, percentage of verbs and nouns, Stanford dependencies) and *sentiment* (terms with their polarity). The outcomes resulted to the final decision about the feature sets that have been preferred in the next section.

6.2.1 First statistical results

In the first place, I did not use any machine learning techniques and I just collected some first statistical results regarding the reviews of the two conferences like the review length per score, the percentage of adjectives etc. Table 7 and Table 8 present these results for the reviews from KDD and ECML respectively.

I start the analysis of the results based on the similarities between the two conferences. According to Figure 10 and the last column of tables 7 and 8, which show the average and standard deviation of different components, we observe that except for review length all of the other components are contiguous between the two conferences. More specifically, reviewers use similar percentage of adjectives nouns, adverbs and verbs, numbers and valence shifters. A closer look shows that for both conferences the higher the score the higher the fraction of adjectives. Regarding the polarity, which was estimated by SentiWordNet lexicon, the smaller the score the lower the polarity. Finally, for both conferences reviewers use overstatements more frequent than understatements.

On the contrary, reviews from ECML are longer than reviews from KDD (251.94 as opposed to 195.80). Reviewers from KDD use more negative words when they want to reject a paper. On the other hand, reviewers from ECML are not so strict with the papers they reject and even for the reviews belonging to class D, which corresponds to strong reject, SentiWordNet lexicon revealed positive polarity. Similarly, reviewers from KDD that strongly accept a paper use words that reveal more positive opinion in contrast to reviewers from ECML where the polarity is lower. For both conferences, polarity increases linearly as far as the score increases.

Table 7 Statistical results regarding reviews from research papers submitted to KDD.

SCORE	1	2	3	4	5	6	Average±STD
Review length	212.84	231.47	213.55	197.62	163.93	155.36	195.80±30.11
adjectives	5.95%	6.56%	6.85%	6.95%	7.51%	8.01%	6.97±0.72%
adverbs	6.52%	7.02%	6.93%	6.57%	6.36%	6.50%	6.65±0.26%
verbs	14.41%	14.00%	14.20%	13.99%	13.19%	12.84%	13.77±0.62%
nouns	19.54%	21.11%	21.05%	20.60%	20.72%	19.69%	20.45±0.68%
overstatements	3.52%	3.87%	3.73%	3.48%	3.49%	3.15%	3.54±0.25%
understatements	2.85%	2.62%	2.75%	2.81%	2.89%	2.13%	2.68±0.28%
numbers	1.81%	2.22%	2.42%	2.44%	2.18%	2.48%	2.26±0.25%
polarity[5]	-0.15	0.23	0.33	0.41	0.65	1.22	-

[5] SentiWordNet lexicon has been used in order to assign a polarity value to each review.

Table 8 Statistical results regarding reviews from research papers submitted to ECML.

SCORE	D	C	B	A	Average±STD
Review length	298.14	284.95	222.40	202.25	251.94±46.78
adjectives	6.81%	7.12%	7.46%	8.36%	7.44±0.67%
adverbs	7.87%	7.22%	7.10%	6.01%	7.05±0.77%
verbs	13.76%	13.82%	13.75%	13.52%	13.71±0.13%
nouns	21.74%	21.57%	21.43%	21.49%	21.56±0.13%
overstatements	3.59%	3.67%	3.62%	3.97%	3.71±0.17%
understatements	2.58%	2.93%	2.90%	2.42%	2.71±0.25%
numbers	2.19%	2.32%	2.08%	1.59%	2.05±0.32%
polarity[5]	0.28	0.62	1.07	1.12	-

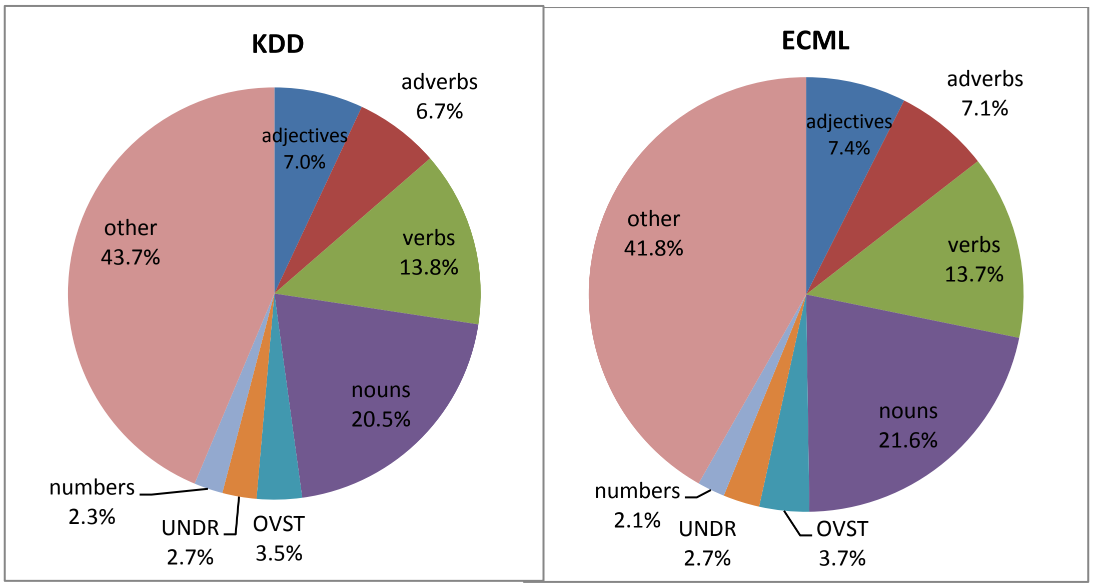


Figure 10 Statistical results per conference.

6.2.2 Simpler problems

To further understand the two corpora and realize the difficulty of extracting opinions I trained MNB classifier using two different feature sets (unigrams and bigrams), in order to solve theoretically simpler and also more difficult problems. Assuming we use reviews from KDD; separating reviews from papers with a strong reject (score 1) from those with a strong accept (score 6) is theoretically a simpler problem than separating reviews from papers with a weak reject (score 3) from those with a weak accept (score 4). We expect that reviewers in the first case use completely different vocabulary while, in the latter case the vocabulary is more similar. The same assumption can be transferred to ECML. In addition to that, in this section I investigate which scores of KDD correspond to scores of ECML and vice versa.

Table 9 shows the accuracy percentage for different tasks. More specifically, the first column corresponds to the different binary classification problems. For KDD 12vs56 means that reviews with scores 1, 2 are treated as negative and reviews with scores 5, 6 as positive, while reviews with scores 3,4 have been excluded. Similarly, for ECML D, C are treated as negative and B, A as positive. The rest of the columns show the performance of every model when evaluated on different domains (columns with labels KDDtoKDD, KDDtoECML, ECMLtoECML, ECMLtoKDD). Finally, columns with the R label rank the models according to their performance within the corresponding domain. The reader can refer to this column in order to find easier the more or less difficult problems.

The results really support the assumptions made in the beginning of this subsection. In-domain evaluation reveals that the problem that can easier be learned is the classification of reviews with scores 1 versus 6 and D versus A, for KDD and ECML respectively. For KDD MNB with unigrams successfully classifies the reviews to the two categories and reaches 75.49% of correct classification. Using bigrams led to slightly worse results and accuracy of 75.22%. For ECML the performance of MNB is even better independent of the feature set (81.65%). At the same time the most difficult problem is to separate reviews with scores 3 versus 4 and C versus B, which reaches the last position in the rank.

Table 9. Results regarding different kinds of problems. Percentage of accuracy and rank of the different problems when MNB is trained using unigrams or bigrams and evaluated on each conferences combination.

model	KDD to KDD	R	KDD to ECML	R	ECML to ECML	R	ECML to KDD	R
Bigrams 12vs56 (DCvsBA)	73.6%	5.5	65.89%	6.0	69.93%	6.0	72.86%	5.0
Bigrams 12vs56 (DvsA)	73.6%	5.5	81.81%	1.0	81.65%	2.5	74.13%	1.0
Bigrams 1vs6 (DvsA)	75.22%	2.0	79.35%	2.0	81.65%	2.5	73.16%	4.0
Bigrams 3vs4 (CvsB)	57.96%	9.0	58.09%	9.0	62.67%	9.0	58.52%	10.0
Unigrams 12vs56 (DCvsBA)	74.15%	3.5	63.0%	8.0	70.09%	5.0	72.1%	6.0
Unigrams 12vs56 (DvsA)	74.15%	3.5	78.01%	3.0	81.65%	2.5	73.69%	2.0
Unigrams 1vs6 (DvsA)	75.49%	1.0	77.26%	4.0	81.65%	2.5	73.31%	3.0
Unigrams 3vs4 (CvsB)	57.74%	10.0	56.84%	10.0	58.14%	10.0	58.92%	9.0
Bigrams 123vs456 (DCvsBA)	67.62%	7.0	66.5%	5.0	65.52%	8.0	69.93%	8.0
Unigrams 123vs456 (DCvsBA)	65.94%	8.0	64.76%	7.0	66.01%	7.0	70.09%	7.0

Regarding the out-of domain evaluation, the highest accuracy corresponds to the model trained using bigrams and reviews with scores 12vs56 and applied to reviews with scores DvsA. Similarly, when MNB is trained and applied from ECML to KDD the highest accuracy is achieved by the reverse process (using bigrams and reviews with scores DvsA, applying to reviews with scores 12vs56). The most interesting outcome is that the in-domain performance is equivalent to the out-of domain performance both for KDD and ECML. Training the model on ECML and applying it to KDD leads to slightly better performance compared to the in-domain one from KDD to KDD (74.13% as opposed to 73.6% respectively). The same results stand from KDD to ECML. This is the first evidence that the two conferences have similarities and traditional machine learning approaches can be used both for in-domain and out-of domain learning. It also shows that reviews with scores 1,2 and 5,6 correspond to reviews with scores D and A respectively. In contrast to that, accuracy percentage show that reviews with score 1 do not correspond to reviews with score D and reviews with score 6 do not correspond to reviews with score A.

Finally, the outcomes show that ECML can easier been learned compared to KDD. Accuracy, achieved by the models when they are trained and evaluated on ECML is higher compared to

KDD for all of the problems examined in this section. However, this outcome is further investigated in the next section.

6.2.3 Common patterns

Similar to Kim et al. (2006) different classes of features are analyzed that are most possible to capture opinions. Until now we have seen experiments using n-grams. The findings indicate that the performance of unigrams and bigrams depends on the learning problem and the learning algorithm as a result; I could not recommend one of them as more possible to capture opinions. In this section I investigate if different patterns like specific POS and Stanford dependencies are more likely to reveal opinions.

More specifically, as suggested by Turney (2002) I extracted pairs of adjacent words that follow specific pattern. I also created feature sets that only include isolated words of different POS like adjectives (JJ), nouns(NN) and adverbs(RB) as researchers maintain that some of them are more likely to reflect opinions. As baseline I use the whole review tagged with the POS (Pang et al. 2002). Finally, I experimented with Stanford dependencies in order to find patterns based on them. The baseline is the whole review with all of the Stanford dependencies as extracted by the parser. MNB has been trained in order to solve binary classification problems, where reviews with scores 1, 2, 3 (D, C) have been treated as negative and reviews with score 4, 5, 6 (B, A) have been treated as positive. Reader can refer to APPENDIX C, which explains the abbreviations and shows the accuracy percentage and the average ranks regarding the in-domain and out-of domain evaluation for all of the extracted patterns as presented on the figures below.

I base the discussion of the results on the average ranks of the different models, as recommended by Demsar (2006). The figures below show the feature sets that capture more opinions based on the average rank of each pattern regarding the in-domain and out-of domain evaluation. Firstly, we observe that the baselines outperform the extracted patterns both when the models are evaluated within the domain they are trained and when they are evaluated on a different domain. Secondly, regarding the POS patterns, JJ and RB offer more information compared to other POS as they appeared higher in the rank both within the same domain and across domains. On the other hand, NNs capture opinions within domains but they are less informative when they are used across domains. This is something expected as nouns are domain dependent and the two conferences focus on different areas of machine learning. JJ_NN which is a pattern known for revealing opinions is not so informative here for any of the conferences while, RB_VB can achieve a tradeoff between learning and transferring. Different types of Stanford dependencies capture the same amount of opinions both when evaluated within a domain and when they are applied across domains. MOD is the most informative, while the worst results were that of COMP.

While different patterns capture different amounts of opinions the analysis of the results shows that using the whole review annotated by the POS or Stanford dependencies gives more information than isolated patterns. This observation resulted to the final decision about the feature sets that have been preferred in the next section.

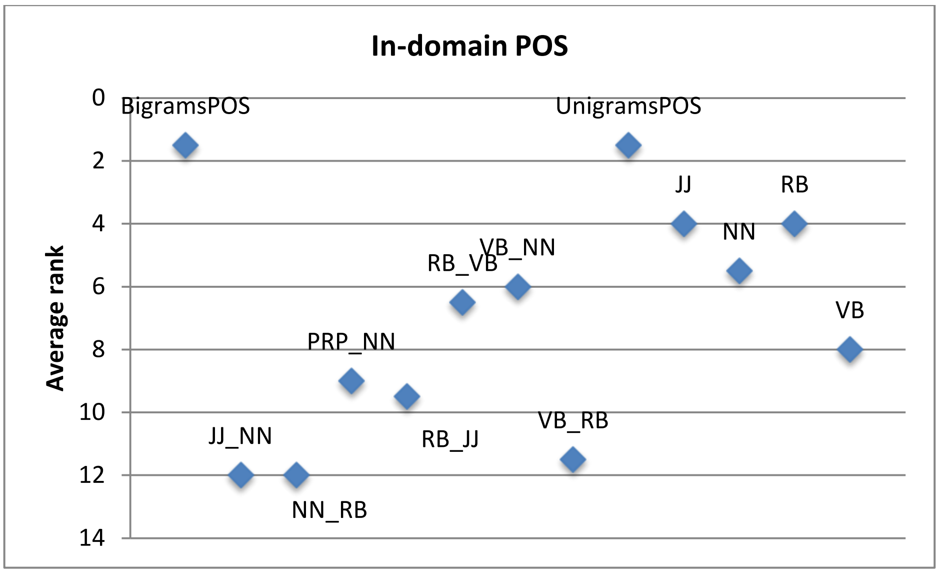


Figure 11 Ranking of the POS that are more likely to capture opinions when evaluated within the same domain (KDDtoKDD or ECMLtoECML).

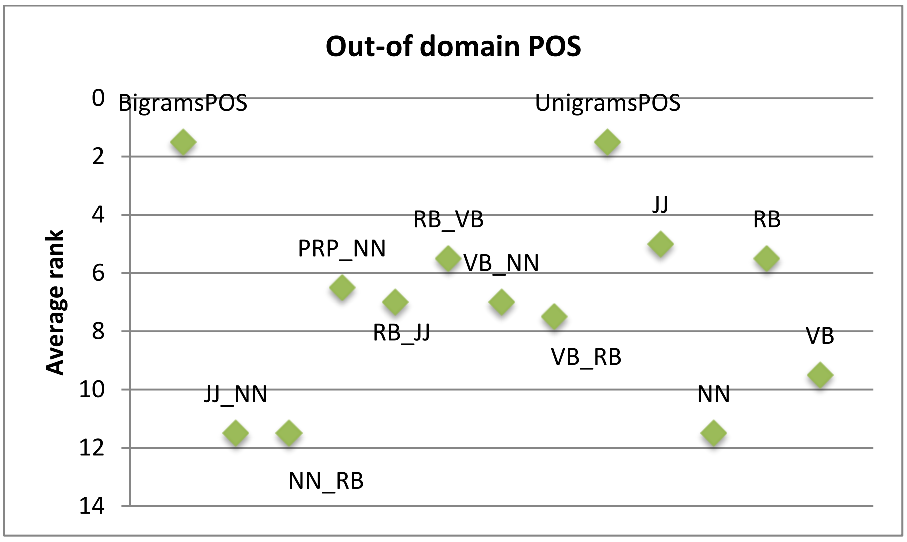


Figure 12 Ranking of the POS that are more likely to capture opinions when source and target domain are different (KDDtoECML or ECMLtoKDD).

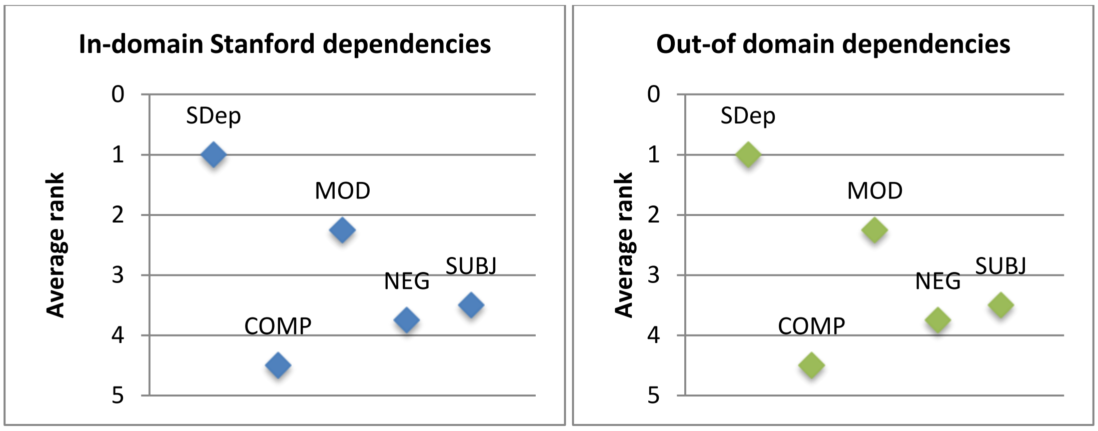


Figure 13 Stanford dependencies that are more likely to capture opinions when evaluated within the same domain (left figure) when source and target domain are different (right figure).

6.3 Extract opinions, apply transfer learning

In this section machine learning algorithms have been trained and evaluated, using different feature sets, in order to solve and transfer the three learning problems as described in sections 2.2 and 4.2. The outcome of this research completes the aims and objectives regarding the project, answers many research questions and gives evidence regarding the similarity between the reviews of ECML and KDD. In the previous section we saw that different patterns capture different amounts of opinions, but using the whole review annotated by the POS or Stanford dependencies gives more information than isolated patterns. This observation resulted to the final decision about the feature sets that have been preferred in this section and include, unigrams, bigrams, unigrams tagged with POS, bigrams tagged with POS, all the extracted Stanford dependencies, unigrams tagged with POS and replaced by valence shifters, bigrams tagged with POS and replaced by valence shifters.

Experiments have been performed in a common way, independent of the learning problem. The whole process has been described analytically in section 5 but it is repeated briefly: Machine learning algorithms are trained using different feature sets. Similar to Aue & Gamon (2005) every trained model is tested both on its own domain and on the domain of the alternative conference. Models are trained on reviews from KDD and evaluated on KDD (KDDtoKDD: In-domain evaluation). These models are also applied and evaluated on reviews from ECML (KDDtoECML: Out-of domain evaluation). The same process is repeated on the opposite directions (ECMLtoECML: In-domain evaluation, ECMLtoKDD: Out-of domain evaluation).

For every learning problem I analyze the results in order to find the most competitive models:

1) The best in-domain model separately for each conference.
2) The best model when transfer learning is applied (KDDtoECML and ECMLtoKDD).
3) The out-of domain models that are equivalent to the corresponding in-domain.

Regarding the third case: Firstly, a model is trained and tested on the one conference (for example ECMLtoECML). Secondly, the same algorithm and feature set is used in order to build a model using the reviews from the alternative conference (for example the model is trained on KDD and tested on ECML (KDDtoECML)). The performance of the two models is compared. If the in-domain performance is higher than the out-of domain performance, two-tailed t-test is used, as discussed in section 5.3.2, in order to further investigate if the difference is statistically significant. If the p-value is lower than 0.05, which shows that the difference is significant, I assume that transfer learning cannot be applied otherwise the difference is not significant and applying this model to a different domain is successful. In addition to that, I assume that transfer learning fails when the performance is below the baseline or when the model completely fails to predict one or more classes (weighted recall is equal to 0). Similarly, in-domain learning fails when a model performs lower than the baseline or when the model completely fails to predict one or more classes (weighted recall is equal to 0 for).

In the beginning of each section the first table shows the models that have been trained for this learning problem. A second table, which consists of nine columns, contains the results regarding the performance of every model. The first column corresponds to the ids of the models. The rest of the columns show the performance of every model when evaluated on different domains (columns with labels KDDtoKDD, KDDtoECML, ECMLtoECML, ECMLtoKDD). Finally, columns with the R label rank the models according to their performance within the corresponding domain. The reader can refer to this column in order to find easier the more or less competitive models. APPENDIX F contains results from different statistical tests that have been performed during this section.

6.3.1 Classification

The first problem under investigation is the opinion classification. Different algorithms using different feature sets have been trained, in order to find which model can better classify every review into two or more categories based on the overall opinion expressed by the authors. I experimented with sever different feature sets and two different machine learning algorithms, which resulted in fourteen models as presented in Table 3. The first model is used as the baseline (majority class). The optimized parameters for SMO regarding the binary classification problem have been presented in section 6.1.3. The same process of finding the best parameters was repeated for multiclass classification and similar results were obtained. For this reason during the classification problem SMO is used with linear kernel and complexity parameter equal to 0.1.

Table 11, Table 12 and Table 15 show the percentage of accuracy and the rank of the competing models for all domain combinations regarding the binary classification, three class classification and four class classification respectively. The models that could be applied to a different domain without having statistically significant difference with the corresponding in-domain models are underlined. Question mark corresponds to models that perform better than the baseline but completely fail to predict one or more classes (weighted recall is equal to 0).

Binary Classification

I started the experiments from the simplest scenario, which is to distinguish reviews of papers that have been accepted and consequently treated as having positive opinion, from those that have been rejected and treated as having negative opinion. Within KDD MNB – Sdep reaches the highest accuracy (67.99%), while within ECML MNB – Unigrams (70.09%). For both domains all of the models perform better than the baseline. These first results show that binary classification is not a difficult problem and different feature sets could probably lead to even better results.

The model trained on KDD and applied to ECML with the highest accuracy is MNB – BigramsPOS (67.82±0.39%). However, its performance is lower compared to the in-domain performance of the same model within ECML (68.24±3.65%). This difference was calculated as statistically significant and indicates that transfer learning fails. For the opposite process the model trained on ECML and applied to KDD with the highest accuracy is MNB – Sdep. The performance of this model again is lower compared to the in-domain within KDD but this time the difference is not statistically significant, which indicates that transfer learning from ECML to KDD is successful. I further investigate if this model can be applied from KDD to ECML and statistical test shows that it really can. Added to this, I used paired t-test in order to see if the difference between MNB-Sdep and MNB-BigramsPOS from KDD to ECML is statistically significant. The p-value was higher than 0.05 which shows that the two models are equivalent even if MNB-BigramsPOS is higher in the rank. It should be noted that MNB-Sdep is the only model that can be transferred on both directions. At the same time, there are models that can be applied from KDD to ECML but not the opposite (from ECML to KDD) and vice versa. Finally, SMO could never been transferred independent of the feature set.

Regarding the two learning algorithms, MNB and SMO perform equally within the same domain if averaged over all feature sets[6]. Furthermore, regarding the feature sets, for KDD any feature set that consists of pairs of words (tagged with more information or not) outperform those that consist of isolated words. On the contrary, in-domain evaluation for ECML reveals that models perform similarly independent of the feature set. For example, feature sets of valence shifters do not really capture more opinions compared to extracting

[6] I performed the Wilcoxon signed ranks test between these two algorithms. The test gave a p-value higher than 0.05, indicating not a significant difference with 95% certainty. (p-value for KDD = 0.5781, p-value for ECML = 0.1094)

simple Unigrams or Bigrams. Across domains sometimes Bigrams-pos and sometimes Stanford dependencies capture more opinions. However, the only feature set that could be transferred for both domains was Stanford Dependencies.

To sum up, ECML is a simpler dataset than KDD as all of the models achieved higher accuracy. All of the models presented equivalent performance when trained and evaluated on ECML (paired-t test between MNB - Unigrams and all of the other models has been performed). On the other hand, for KDD models trained using bigrams performed significantly better than models trained using unigrams (paired-t test between MNB - Sdep and all of the other models has been performed). The final recommendation regarding the binary classification is MNB – Sdep both for the in-domain classification and to apply transfer learning to the alternative conference.

Table 10 Machine learning algorithms and feature set combinations regarding the classification problem.

Model id	**Model description (Machine learning algorithm - feature set)**
m1	Baseline – Majority class
m2	MNB - bigramsPOS
m3	MNB - bigramsPOSVal
m4	MNB - SDep
m5	MNB - unigramsPOS
m6	MNB - unigramsPOSVal
m7	MNB - Bigrams
m8	MNB - Unigrams
m9	SMO - bigramsPOS
m10	SMO - bigramsPOSVal
m11	SMO - SDep
m12	SMO - unigramsPOS
m13	SMO - unigramsPOSVal
m14	SMO - Bigrams
m15	SMO - Unigrams

Table 11 Results regarding the binary classification problem. Percentage of accuracy and rank of the competing models when trained and evaluated on each conferences combination.

model	**KDDtoKDD**	**R**	**KDDtoECML**	**R**	**ECMLtoECML**	**R**	**ECMLtoKDD**	**R**
m1	64.76±0.42%	15.0	61.67±0.0%	15.0	61.67±0.25%	15.0	64.76±0.0%	7.5
m2	67.13±3.05%	5.0	67.82±0.39%	1.0	68.24±3.65%	9.5	64.76±0.53%	7.5
m3	67.19±2.31%	4.0	66.29±0.64%	4.0	68.65±4.3%	6.0	63.68±0.61%	12.0
m4	67.99±3.05%	1.0	66.54±0.45%	2.0	69.45±4.35%	3.0	66.77±0.48%	1.0
m5	66.19±3.26%	10.0	64.37±0.52%	9.0	68.65±2.41%	6.0	65.18±0.36%	4.0
m6	66.31±4.02%	9.0	63.38±0.55%	14.0	67.28±2.24%	12.5	64.84±0.44%	6.0
m7	67.62±2.95%	3.0	66.5±0.55%	3.0	69.93±4.24%	2.0	65.52±0.77%	3.0
m8	65.94±4.0%	12.0	64.76±0.74%	6.0	70.09±2.47%	1.0	66.01±0.63%	2.0
m9	66.75±2.29%	6.5	64.07±0.94%	11.0	67.28±3.67%	12.5	63.83±0.41%	11.0
m10	66.13±3.63%	11.0	63.52±0.73%	12.0	66.89±4.19%	14.0	63.37±0.39%	13.0
m11	65.63±2.11%	13.5	64.59±0.65%	7.0	68.01±4.06%	11.0	63.95±0.7%	9.0
m12	66.75±2.58%	6.5	64.87±0.84%	5.0	69.21±4.55%	4.0	63.93±0.78%	10.0
m13	66.69±2.15%	8.0	64.42±1.03%	8.0	68.65±4.02%	6.0	62.76±0.4%	15.0
m14	67.8±3.32%	2.0	63.47±0.66%	13.0	68.32±4.04%	8.0	65.08±0.85%	5.0
m15	65.63±4.39%	13.5	64.12±0.89%	10.0	68.24±3.25%	9.5	62.95±0.64%	14.0

Three-class classification

The next step was to classify the reviews into three different categories, negative, neutral and positive. Within KDD SMO – Unigrams reaches the highest accuracy, at the same time all of the models except two perform better than the baseline. Within ECML MNB - Unigrams reaches the highest accuracy; however, a closer look on the results shows that the learning fails. As discussed in section 5.3.1 for multiclass classification we do not only look on the accuracy but also on the weighted recall per class, which reveals those classes that the classifier performs better or worse. As a result for the latter model, recall was high for the classes negative and neutral but zero for the class positive. The model has learned to classify correct the negative and neutral reviews, however, this has resulted to also classify as negative or neutral all of the positive reviews. Positive reviews have been misclassified for all of the models trained and evaluated on ECML and not only for the first in the rank. So, learning the three class problem completely fails for ECML.

Furthermore, all of the models trained on KDD and applied to ECML fail as they perform lower than the baseline. Thus, just as before we could not find a model in order to learn the three-class classification problem for ECML. On the other hand, SMO - unigramsPOS reaches the highest accuracy when it is trained on ECML and applied to KDD. The performance of this model is slightly better compared to the corresponding model within KDD and transfer learning on this direction is successful. We observe that even if a model is not able to succeed within its native domain, it can apply transfer learning to a different domain.

Regarding the two learning algorithms, similarly to the binary classification MNB and SMO perform equally within the same domain if averaged over all feature sets. A different outcome now is that MNB could never been transferred independent of the feature set but SMO could be transferred for some of the feature sets. This is the opposite compared to the binary classification. It seems that SVMs are more robust, as far as the problem becomes more difficult. Regarding the feature sets, those that consist of unigrams perform slightly better compared to those that consist of bigrams if they are tested within the same domain. Feature sets of unigrams are appeared into the top six of the rank. No conclusions could be reached regarding the feature set when the models are applied to a different domain.

But why learning the three-class classification problem is impossible for ECML? Which are those classes that the models find difficult to predict? In order to answer these questions, I calculated the confusion matrices for SMO-Unigrams model when trained and evaluated on KDD and for MNB-Unigrams when trained and evaluated on ECML (Table 13 and Table 14 respectively). Misclassifications mainly exist between the adjacent classes. More specifically, for both conferences the misclassifications converge toward the neutral class, which in this case is the majority class. Negative reviews are classified as neutral and almost never as positive. On the other hand, for KDD positive reviews are also mainly classified as neutral, but quiet enough are also classified as negative, while for ECML all of the positive reviews are classified as neutral and only two are classified as negative. These results are in accordance with the challenges discussed in section 4.3 and probably correspond to the fact that reviewers make negative comments even if they want to accept a paper. Similarly, for negative reviews they write positive comments but in this case negative comments always outnumber, which results to having misclassifications only between negatives and neutrals and not between negatives and positives. While this explanation seems reasonable, bear in mind that the models probably have been affected by the fact that the classes are imbalanced. In the future it would be interesting to investigate if this is a problem. A useful source of information can be found in Chawla et al. (2004) where numerous methods that deal with the imbalance-class problem are summarized.

To sum up, three class classification is a more difficult problem compared to binary classification. This is obvious by the lowest accuracy of all models and also by the fact that learning completely fails for the reviews submitted to ECML. Only KDD could be learned and probably conferences that have a smaller score range are more difficult to distinguish neutral reviews. However, this is just an observation and more experiments are needed.

Table 12 Results regarding the three-class classification problem. Percentage of accuracy and rank of the competing models when trained and evaluated on each conferences combination

model	**KDDtoKDD**	**R**	**KDDtoECML**	**R**	**ECMLtoECML**	**R**	**ECMLtoKDD**	**R**
m1	54.9±0.36%	13.0	68.97±0.0%	1.0	68.97±0.27%	8.5	54.9±0.0%	13.0
m2	55.27±3.54%	10.0	68.2±0.51%	2.5	69.05±0.98%	6.5**?**	54.47±0.25%	15.0
m3	55.52±4.48%	9.0	68.2±0.53%	2.5	69.37±0.45%	4.5**?**	54.85±0.23%	14.0
m4	55.15±2.11%	11.0	66.29±0.38%	8.0	68.17±1.66%	14.5	55.47±0.37%	9.0
m5	57.13±2.97%	2.0	67.47±0.75%	5.0	69.05±1.46%	6.5**?**	55.67±0.34%	8.0
m6	57.07±2.9%	3.0	67.11±0.76%	7.0	69.61±1.22%	2.5**?**	55.43±0.38%	11.0
m7	55.58±2.53%	8.0	68.15±0.25%	4.0	69.37±1.49%	4.5**?**	54.93±0.31%	12.0
m8	56.01±3.13%	5.0	67.19±0.27%	6.0	70.17±2.39%	1.0**?**	55.73±0.36%	7.0
m9	54.96±3.05%	12.0	62.62±0.87%	13.0	69.61±2.56%	2.5**?**	55.93±0.38%	5.0
m10	54.4±3.52%	14.0	62.37±0.88%	14.0	68.57±2.32%	11.0	55.91±0.4%	6.0
m11	53.72±2.4%	15.0	60.58±0.86%	15.0	68.41±2.18%	12.0	55.44±0.39%	10.0
m12	55.71±2.55%	7.0	63.64±0.66%	10.0	68.97±2.18%	8.5**?**	56.9±0.52%	1.0
m13	55.77±2.63%	6.0	63.1±0.75%	11.0	68.17±2.17%	14.5	56.0±0.22%	3.0
m14	56.27±3.14%	4.0	62.85±0.61%	12.0	68.33±2.63%	13.0	55.99±0.29%	4.0
m15	57.94±3.21%	1.0	65.53±0.88%	9.0	68.72±1.97%	10.0	56.4±0.29%	2.0

Table 13 Confusion matrix for model m15 when trained and evaluated on KDD.

a	**b**	**c**	**← classified as**
188.0	325.0	1.0	**a = Negative**
137.0	740.0	8.0	**b = Neutral**
32.0	175.0	6.0	**c = Positive**

Table 14 Confusion matrix for model m8 when trained and evaluated on ECML.

a	**b**	**c**	**← classified as**
45.0	251.0	0.0	**a = Negative**
29.0	830.0	1.0	**b = Neutral**
2.0	89.0	0.0	**c = Positive**

Four-class classification

Finally, classifiers are built that classify the reviews into four different categories, strong negative, negative, positive and strong positive. Within KDD MNB – Unigrams reaches the highest accuracy (40.63±2.76%), at the same time all of the models perform better than the baseline. Within ECML SMO-BigramsPOS reaches the highest accuracy (46.43±4.32%). All of the models perform better than the baseline however; again not all of them are able to predict all of the classes. Similar to the previous problem models m2, 3, 5, 6, 7, 8 and 11 fail to classify reviews that belong to the strong positive class (the weighted recall is equal to zero). Regardless of these models, learning the four-class classification problem is possible for both conferences.

When we apply the models from KDD to ECML only four models perform better than the baseline. The best models is MNB-Bigrams, which can apply transfer learning, in contrast to the case that it could not been learned within ECML. On the other hand, SMO – Unigrams is also at the top of the rank but it has significantly lower accuracy compared to the in-domain performance of the corresponding model so the model could not really be transferred. When we apply the models from ECML to KDD all of them except one perform better than the baseline. It should be noted that the four models that are higher in the rank have weighted recall for class strong accept equal to zero, so we assume that they fail to transfer learning. As a result, the best model when we apply transfer learning is MNB – UnigramsPOS. However, its performance is lower compared to the in-domain performance of the corresponding model when trained and evaluated on KDD. This difference was calculated as statistically significant and indicates that transfer learning fails. We conclude that transferring the four-class problem is possible from KDD to ECML but not from ECML to KDD. However, even in this case the classifier barely outperforms the baseline.

Regarding the two learning algorithms, similar to the binary and three-class classification MNB and SMO perform equally within the same domain if averaged over all feature sets. We do not have any conclusions when these algorithms are used to transfer learning as only two models could really been transferred. Regarding the feature sets unigrams perform slightly better compared to bigrams, if they are evaluated within KDD. Feature sets of unigrams are appeared into the top 5 of the rank. No conclusions could be reached regarding the feature set when the models are trained and applied to ECML or applied to a different domain.

A closer look on the results shows that low performance occurs due to the similarity between the adjacent classes. Table 16, Table 17, Table 18 and Table 19 show the confusion matrices for models m8 (KDDtoKDD), m4 (ECMLtoKDD), m9 (ECMLtoECML) and m7 (KDDtoECML) respectively. Similar to the three-class classification problem misclassifications mainly exist between the adjacent classes. More specifically, all models have better learned to classify the negative class, which is the majority class. The misclassifications converge toward the mean classes. Strong negative reviews almost never are classified as strong positive. On the other hand, strong positive reviews frequent are classified as strong negative. Similar conclusions with the three-class classification scenario, regarding the way the reviewers express their opinions and the imbalanced-class problem should be considered here as well.

To sum up, four class classification is a more difficult problem than binary classification. This is obvious by the lowest accuracy for all models. ECML is a simpler dataset than KDD as all of the models achieved higher accuracy. Within KDD, MNB – Unigrams reaches the highest accuracy (40.63±2.76%), and this is statistically significant compared to some of the competing models, but not all. Within ECML, SMO-BigramsPOS reaches the highest accuracy (46.43±4.32%). Transfer learning is possible only from KDD to ECML and the only model that could be transferred was MNB-Bigrams.

Table 15 Results regarding the four-class classification problem. Percentage of accuracy and rank of the competing models when trained and evaluated on each conferences combination

model	KDDtoKDD	R	KDDtoECML	R	ECMLtoECML	R	ECMLtoKDD	R
m1	32.88±0.09%	15.0	37.93±0.0%	4.0	37.93±0.48%	15.0	32.88±0.0%	14.0
m2	36.91±2.39%	9.5	37.43±0.75%	6.0	44.59±3.76%	4.5**?**	34.04±0.61%	10.0
m3	36.91±3.08%	9.5	36.51±0.47%	8.5	44.59±2.85%	4.5**?**	33.67±0.7%	11.0
m4	37.22±2.65%	7.0	35.28±0.62%	14.0	42.75±3.25%	13.0	35.14±0.4%	5.0
m5	39.08±3.25%	3.0	35.69±0.66%	11.0	43.86±2.18%	6.5**?**	33.22±0.48%	13.0
m6	39.2±3.41%	2.0	35.67±0.48%	12.0	43.23±2.84%	11.0**?**	33.32±0.45%	12.0
m7	37.78±1.63%	6.0	38.15±1.05%	1.5	45.47±4.03%	2.0?	34.52±0.48%	8.0
m8	40.63±2.76%	1.0	37.97±0.6%	3.0	43.86±3.46%	6.5**?**	35.01±0.57%	6.0
m9	36.1±3.55%	12.0	36.54±0.56%	7.0	46.43±4.32%	1.0	65.62±0.51%	1.0**?**
m10	35.29±2.96%	13.0	36.43±0.6%	10.0	43.62±4.64%	8.0	64.86±0.45%	2.0**?**
m11	34.92±2.92%	14.0	34.77±0.97%	15.0	41.22±3.05%	14.0**?**	32.85±0.53%	15.0
m12	37.03±1.73%	8.0	37.66±0.79%	5.0	43.31±4.47%	10.0	63.93±0.78%	3.0**?**
m13	38.27±3.51%	5.0	36.51±0.77%	8.5	43.15±3.52%	12.0	62.76±0.4%	4.0**?**
m14	36.35±3.22%	11.0	35.54±0.77%	13.0	44.74±4.68%	3.0	34.36±0.69%	9.0
m15	38.64±4.28%	4.0	38.15±1.04%	1.5	43.46±4.8%	9.0	34.78±0.49%	7.0**?**

Table 16 Confusion matrix for model m8 when trained and evaluated on KDD.

a	b	c	d	←classified as
279.0	188.0	30.0	17.0	**a = Strong negative**
167.0	306.0	35.0	22.0	**b = Negative**
105.0	190.0	48.0	12.0	**c = Positive**
60.0	94.0	37.0	22.0	**d = Strong positive**

Table 17 Confusion matrix for model m4 when trained on ECML and evaluated on KDD.

a	b	c	d	←classified as
71.0	342.0	100.0	1.0	**a = Strong negative**
39.0	350.0	140.0	1.0	**b = Negative**
15.0	227.0	113.0	0.0	**c = Positive**
12.0	106.0	91.0	4.0	**d = Strong positive**

Table 18 Confusion matrix for model m9 when trained and evaluated ECML.

a	b	c	d	←classified as
88.0	163.0	45.0	0.0	**a=Strong negative**
64.0	302.0	107.0	0.0	**b = Negative**
24.0	174.0	188.0	1.0	**c = Positive**
4.0	38.0	48.0	1.0	**d = Strong positive**

Table 19 Confusion matrix for model m7 when trained on KDD and evaluated on ECML.

a	b	c	d	←classified as
158.0	133.0	5.0	0.0	**a = Strong negative**
189.0	266.0	18.0	0.0	**b = Negative**
137.0	210.0	33.0	7.0	**c = Positive**
27.0	44.0	13.0	7.0	**d = Strong positive**

6.3.2 Regression

For this learning problem I trained four different regression algorithms for each domain, using two different feature sets, which resulted in eight models as presented in Table 20. I experimented only with simple unigrams and bigrams as other feature sets did not really offer more information during the three and four-class classification. It should be remarked, that AR and Bag algorithms are trained using the 1000 unigrams or bigrams with the highest information gain. Regression is a time consuming method and there was a need to reduce the feature space. In contrast, SMOreg was trained, similar to classification, without any feature space reduction because it can deal with high dimensions by performing very fast at the same time.

Similar to classification the first aim was to find the best parameters for SMO regression model. The best combinations were SMO with linear kernel and complexity parameter equal to 0.001. I refer to this model as SMOreg. Equivalent performance was presented by SMO with RBF kernel, complexity parameter equal to 10 and gamma 0.005. I refer to this model as SMOrbf.

In the same manner as classification, I started from theoretically simpler problems and I continued with more difficult. For the first round of experiments, I merge reviews with different scores in order to have a three score range for both conferences. Scores have been merged as described in the beginning of this section. I will refer to this problem as *the three-score regression problem*. After that I merge scores in order to have a four score range for both conferences and this is the *four-score regression problem*. Finally, I do not merge anything, which results to the *full-score regression problem*.

I avoided presenting the performance of all of the models separately for every sub problem as it would be meaningless for the reader. I start the analysis of the results from some interesting observations and after that I recommend the models that perform best on predicting numerical scores. Reader can refer to APPENDIX D, which summarizes all of the results regarding the regression problem.

The first interesting outcome, when regression models are used for numerical prediction, is that all of them outperform the baseline both in cases they are used for in-domain evaluation and when they are applied to a different domain. This shows that regression is a more robust solution. Even if in some cases the models applied to a different domain do not really perform equally with the models trained within the domain, they are able to outperform the baseline, which shows that different feature sets could probably help on the transfer. On the other hand, similar to classification, as far as the problem becomes more difficult, (from three-score regression to full-score regression) the MAE increases and the performance of the models decreases. This difference was calculated as statistically significant. Finally, the results show again that ECML is an easier conference to be learned compared to KDD because the MAE for ECML is always smaller compared to KDD.

Furthermore, models trained on ECML and evaluated on KDD always perform better compared to models trained and evaluated on KDD. A different domain offers more information and this shows that the two conferences have similarities. The only exception is the *full-score regression problem,* where all of the models, both from ECML to KDD and from KDD to ECML, fail to apply to a different domain. In this case, the models outperform the baseline when they are applied to the new domain however, they perform lower than the corresponding models, which are trained and evaluated within the same domain. This difference has been calculated as statistically significant. Even if we use regression models for predicting numerical scores, there still the need of having the same score range between the two conferences.

Experiments performed by Pang & Lee (2006b) show that for the *three-score regression problem* classifiers perform better than regression models and as far as the number of classes increases regression models should be preferred. However, in this research, regarding the *three-score regression problem,* for KDD only a few classifiers performed successfully, while for ECML all of the classifiers fail. We cannot really say that regression models are better compared to classifiers, however, the fact that all of them outperform the baseline gives evidence that they are more sensible for solving numerical prediction problems. Based on bibliography, there is not a straight way to compare regression models with classifiers but my recommendation, regarding these two conferences, would be to use classifiers only for binary classification and regression models in order to predict review scores.

So, which models should be preferred in order to predict numerical scores? I recommend the best model averaged over all the regression sub problems. Table 21 and Table 22 correspond to the results for in-domain and out-of domain evaluation respectively and show the rank of the competing models based on their performance separately for every learning sub problem. The last column shows the average rank. The analysis of the results is based on the average ranks as suggested by Demsar (2006). In-domain evaluation reveals that the best model is SMOrbf – Unigrams. On the other hand, SMOreg – Bigrams lead to slightly lower performance and an overall rank of 2.6 this is in accordance with our knowledge that SVM are robust schemes for predicting numerical scores. AR - 1000Bigrams and Bag - 1000Bigrams do not perform that well, while the worst results are that of the baseline. It is noticed that except for SMOreg the other models perform better if they are trained using unigrams instead of bigrams.

Regarding the out-of domain evaluation, the best overall results are achieved by AR - 1000Unigrams with an overall rank of 2.8. While slightly worse result achieves SMOreg - Unigrams with an overall rank of 3.0. AR was not on the top for in-domain evaluation but it can be used to transfer learning. Ensembles of classifiers are generally known for predicting high performance. Another explanation of this performance is the fact that this algorithm uses the 1000 Unigrams with the higher information gain. This feature set produces a more general model that can be used across the domains. Close to this performance was also SMOrbf – Unigrams, while again all of the models perform better if they are trained using unigrams instead of bigrams.

SMOrbf – Unigrams is the best model regarding the in-domain evaluation. But is this model statistically better compared to the others? Two-tailed paired t-test between this model and all of the competing models revealed that the two main competitors AR – 1000Unigrams and SMOreg - Bigrams are able to perform equally well. On the other hand, AR – 1000Unigrams shows the best performance regarding the out-of domain evaluation but can this model really been applied to a different domain? Two-tailed t-test returns a p-value bigger than 0.05 independent of the direction that transfer learning is applied only when the two conferences have the same score-range[7]. As a result, the difference between the in-domain and out-of domain evaluation is not statistically significant and the model successfully applies transfer learning.

To sum up, similar to classification regression reveals that ECML is an easier conference to be learned and transferred compared to KDD, regression is more meaningful for predicting numerical scores compared to classification and the results seem more promising. Based on the findings, we did not conclude to a specific model that can be used for in-domain and out-of domain learning and achieves the best performance for the both cases. My recommendation would be to use SMOrbf – Unigrams in order to learn within the same domain and AR – Unigrams in order to apply transfer learning to different domains. In the

[7] AR - 1000Unigrams for four-score range: unpaired t-test between KDDtoECML and ECMLtoECML p-value = 0.08360, ECMLtoKDD and KDDtoKDD p-value = 0.39396.

latter case, it should be considered that neither the source nor the target domain will reach the best possible performance but the performance will be equal to the performance if each of them would be tested with models trained on their native domain. Finally, there still the problem of having the same score range between the two conferences in order to be able to apply transfer learning. Models trained and evaluated by using the full score for both conferences completely fail to be applied to a different domain.

Table 20 Machine learning algorithms and feature set combinations regarding the regression problem.

Model id	Model
m1	Baseline – mean score
m2	AR - 1000Bigrams
m3	AR - 1000Unigrams
m4	Bag - 1000Bigrams
m5	Bag - 1000Unigrams
m6	SMOreg - Bigrams
m7	SMOrbf - Bigrams
m8	SMOrbf - Unigrams
m9	SMOreg - Unigrams

Table 21 In-domain evaluation regarding the regression problem. Rank and average rank based on the macroMAE of the competing models when trained and evaluated within the same conference for different regression sub-problems.

model	3KDD toKDD	3ECML toECML	4KDD toKDD	4ECML toECML	6KDD toKDD	Average rank
m1	9.0	9.0	9.0	9.0	9.0	9
m2	8.0	8.0	8.0	6.0	8.0	7.6
m3	7.0	3.0	2.0	3.0	1.0	3.2
m4	5.0	5.0	6.0	8.0	7.0	6.2
m5	6.0	7.0	3.0	5.0	3.0	4.8
m6	2.0	2.0	1.0	4.0	4.0	2.6
m7	4.0	6.0	5.0	7.0	6.0	5.6
m8	3.0	1.0	4.0	1.0	2.0	2.2
m9	1.0	4.0	7.0	2.0	5.0	3.8

Table 22 Out-of domain evaluation regarding the regression problem. Rank and average rank based on the macroMAE of the competing models when trained on one conference and evaluated on a different conference for different regression sub-problems.

model	3KDDto ECML	3ECMLto KDD	4KDDto ECML	4ECMLto KDD	6KDDto ECML	6ECMLto KDD	Average rank
m1	9.0	9.0	9.0	9.0	9.0	9.0	9.0
m2	7.0	6.0	7.0	8.0	8.0	2.0	6.3
m3	1.0	3.0	1.0	5.0	6.0	1.0	2.8
m4	6.0	4.0	6.0	6.0	4.0	4.0	5.0
m5	4.0	8.0	2.0	4.0	3.0	3.0	4.0
m6	5.0	1.0	5.0	1.0	5.0	7.0	4.0
m7	8.0	7.0	8.0	7.0	7.0	8.0	7.5
m8	2.0	5.0	3.0	3.0	2.0	5.0	3.3
m9	3.0	2.0	4.0	2.0	1.0	6.0	3.0

6.3.3 Ranking

Predicting review scores from reviews of research papers is not really required and is not always meaningful. Reviews of research papers always come with a score; added to this, author calibration is a very difficult problem and even if we have the perfect classifier or the perfect regression model, it is possible to fail due to this problem. Last but not least, score ratings are different among different conferences, which make it impossible to transfer regression and classification models across domains without merging scores. In this section, the aim is to train rankers on the reviews of the one conference and apply them to the second conference. But is this problem easier compared to regression and classification? Furthermore, can this problem really be transferred independent of the score range?

This is the final learning problem that has been solved and transferred. As discussed in subsection 5.2 apart from RankNet, which is a probabilistic neural network ranking model, classifiers and regression models have also been used as rankers. This resulted to four different rankers all of them trained using simple unigrams (Table 23).

In the same manner as classification and regression I merge reviews with different scores in order to have a three score range for both conferences. I will refer to this problem as *the three-score ranking problem*. After that I merge scores in order to have a four score range for both conferences and this is the *four-score ranking problem*. Finally, I use the scores as given by the reviewers, this is the *full-score ranking problem*. Reader can refer to APPENDIX E, which summarizes all of the results regarding the ranking problem.

The first difference compared to the two previous learning problems is that as far as the problem becomes more difficult or in other words the score range increases, the performance of the models stays the same and it is not decreased. Rankers can perform the same independent of the score range. A second observation is that when rankers are used to a different domain their performance remains the same independent of the score range. This gives evidence that ranking is an easier problem compared to regression and classification both to be learned and to be transferred. This is in accordance with our prior knowledge according to which, rankers are evaluated on the relative order that can be learned independent of the score range. This property of rankers is very important for reviews of research papers as score ratings can change in different conferences.

So, let us see which models should be preferred in order to rank the reviews and apply transfer learning. Similar to the previous subsection I recommend the best model averaged over all the rank sub problems. Table 24 and Table 25 correspond to the results for in-domain and out-of domain evaluation respectively and show the rank of the competing models based on their performance separately for every learning sub problem. The last column shows the overall rank. Yet again, I analyze the results based on the average ranks as suggested by Demsar (2006). In-domain evaluation reveals that the best model is SMOreg - Unigrams. On the other hand, MNB – Unigrams lead to slightly lower performance and an overall rank of 1.8. AR – 1000Unigrams, while the worst results are that of the RN – 1000Unigrams.

Regarding the out-of domain evaluation the best overall results, once more, are achieved by SMOreg – Unigrams, which reaches the best performance for all of the sub-problems. MNB – Unigrams and AR – 1000Unigrams performed very close to each other, while again the worst results are that of the RN – 1000Unigrams. The low performance of RN was not expected because it has predicted to be a robust ranking scheme. A possible reason is that while the other algorithms have been trained with optimized parameters, this algorithm was trained for suboptimal parameters. More specifically, the neural network was trained for fewer rounds and hidden nodes compared to the recommended (Burges et al. 2005). Unfortunately, the fact that it is time consuming to train neural networks and the limitation of time did not give the opportunity to optimize its parameters.

SMOreg – Unigrams presents the best performance both regarding the in-domain evaluation and the out-of domain evaluation. But is this model statistically significant compared to the others? Can this problem really be transferred if we use this model? Regarding the first question, I performed paired t-test between this model and all of the competing rankers, which showed that the difference in performance is statistically significant only between SMOreg - Unigrams and RN – 1000Unigrams. In addition to that, significance testing performed regarding the transfer learning phase, which shows that this model can really be transferred both from KDDtoECML (p-value is equal to 0.052) and from ECMLtoKDD (p-value is equal to 0.56). For both cases the p-value is higher than 0.05 which shows that the difference with the in-domain evaluation is not statistically significant.

To sum up, similar to the earlier learning problems ranking reveals that ECML is an easier conference to be learned and transferred compared to KDD. Ranking can be transferred independent of the score range. SMOreg – Unigrams was the best ranker both for in-domain and out-of domain learning. However, significance testing revealed that except for RN the other models can be used as rankers equally well.

Table 23 Machine learning algorithms and feature set combinations regarding the ranking problem.

Model id	**Model**
m1	AR - 1000Unigrams
m2	MNB - Unigrams
m3	RN - 1000Unigrams
m4	SMOreg - Unigrams

Table 24 In-domain evaluation regarding the ranking problem. Rank and average rank based on the rank error of the competing models when trained and evaluated within the same conference for different ranking sub-problems.

model	**3KDDto KDD**	**3ECMLto ECML**	**4KDDto KDD**	**4ECMLto ECML**	**6KDDto KDD**	**Average rank**
m1	3.0	3.0	3.0	3.0	2.0	2.8
m2	1.0	1.0	2.0	2.0	3.0	1.8
m3	4.0	4.0	4.0	4.0	4.0	4.0
m4	2.0	2.0	1.0	1.0	1.0	1.4

Table 25 Out-of domain evaluation regarding the regression problem. Rank and average rank based on the rank error of the competing models when trained on one conference and evaluated on a different conference for different ranking sub-problems.

model	**3KDDto ECML**	**3ECMLto KDD**	**4KDDto ECML**	**4ECMLto KDD**	**6KDDto ECML**	**6ECMLto KDD**	**Average rank**
m1	3.0	2.0	3.0	2.0	2.0	2.0	2.3
m2	2.0	3.0	2.0	3.0	3.0	3.0	2.7
m3	4.0	4.0	4.0	4.0	4.0	4.0	4.0
m4	1.0	1.0	1.0	1.0	1.0	1.0	1.0

6.4 Customize Classifiers with generalized features

In the previous section I presented a number of experiments, which performed in order to solve, transfer and compare three different learning problems: classification, regression and ranking. The analysis of the results revealed the best model per learning problem. However, before we are able to find these models we came across with many different categories of them like models that could not learn the problem but could be applied to a different domain and models that could not be applied to a different domain but could learn the problem within a specific domain.

In this section, I focus on binary classification problem and on those models that performed successfully within the domain they were trained, but at the same time they failed to be applied to a different domain. For example, in subsection 6.3.1 MNB trained using Unigrams learns the binary classification problem both for KDD and ECML, can be applied from ECML to KDD, but cannot be applied from KDD to ECML.

The aim of this section is to investigate if alternative approaches can be used in order to transfer models that could not been transferred. As discussed in the beginning of Section 3, machine learning algorithms fail if they are tested on datasets that have different distribution compared to the datasets they were trained. Transfer learning is the area where methods have been developed that deal with this problem.

Aue & Gamon (2005) customize machine learning algorithms with different feature sets in order to see which of them can be used for transfer learning. Similar to their work I customize MNB and SMO with features that have high Frequently Co-occurring Entropy. This is a measure proposed by Tan et al. (2009) in order to extract generalized features. The most of the methods in Transfer learning area focus on finding generalized features between the different domains and they develop methods that use these features.

Frequently Co-occurring Entropy is formalized as:

$$fce_{word} = \log\left(\frac{\mathrm{P_o(word)} * \mathrm{P_n(word)}}{|\mathrm{P_o(word)} - \mathrm{P_n(word)}|}\right), \mathrm{where}$$

$$\mathrm{P}_{domain}(\mathrm{word}) = \frac{\left(\mathrm{N}_{\mathrm{word}}^{domain} + \alpha\right)}{|\mathrm{D}^{\mathrm{domain}}| + 2 * \alpha}\ , domain = n, o,$$

$$N\!: \#of\ times\ the\ work\ occurs\ in\ this\ domain, \qquad D\!: total\ number\ of\ words\ within\ the\ domain$$

In this formula n is referred to the new domain and o is referred to the old domain. So, $\mathrm{P_o(word)}, \mathrm{P_n(word)}$ are the probabilities of the word in the old and in the new domain respectively. This measure returns words that occur frequently between the two domains and at the same time they have similar occurring probability. The measure can be used without need of having labeled data for the target domain.

I experimented with two models that failed to be transferred during the binary classification task: MNB and SMO trained using unigrams. Figure 14 shows the performance of MNB and SMO when it is trained with different cutoffs of generalized features. Default corresponds to the default performance when the algorithms are trained using all of the unigrams of the source domain. CW (Common Words) corresponds to the performance when the algorithms are trained using only the words that are common between the two domains.

MNB – Unigrams could not be transferred from KDD to ECML. SMO – Unigrams could not be transferred neither form ECML to KDD nor from KDD to ECML. For cases that could not be transferred we observe that different cutoffs of generalized features improve the out-of domain performance. On the other hand, MNB – Unigrams could be transferred from ECML to KDD, in this case, different cutoffs of generalize features decrease the out-of domain performance and the default feature set is predicted to be the best. Results show that models that could not be transferred now present improved performance. On the other hand, models that could be transferred perform worse.

More experiments are needed but these first results indicate that simple techniques from Transfer Learning area can be useful in order to transfer models that could not be transferred.

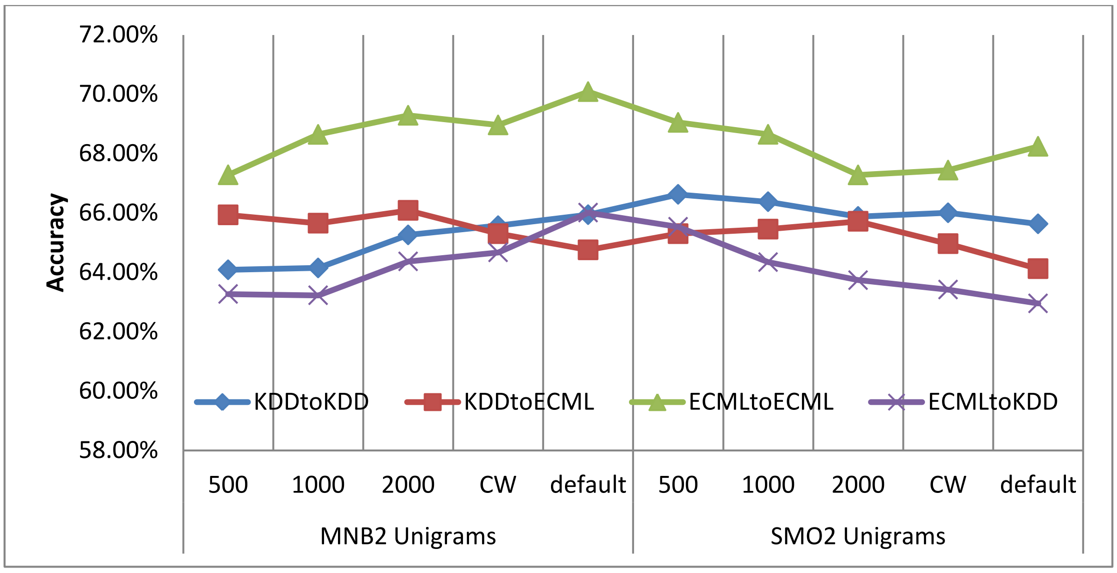


Figure 14 Accuracy percentage of different cutoffs of unigrams with the highest frequently co-occurring entropy. This is a binary classification problem, reviews that have been rejected/accepted are treated as negative/positive.

6.5 Summary

In this section many experiments have been carried out that deal with three opinion mining learning problems when they are applied to reviews of research papers: Classification, regression and ranking. The experimental results have been presented and analyzed, which led to recommendations about the best models within and across domains. The next section summarizes the outcomes, the most interesting similarities between the two conferences and further interprets some observations.

7 Discussion

In the previous section I have examined various methods that can solve and transfer the three different learning problems that were described in Section 2.2. The experiments that were carried out revealed how state-of-the art machine learning algorithms in combination with feature extraction methods can capture opinions in this corpus. The performance of different models has been evaluated, and based on the results, the best model per learning problem and conference was recommended, within and across the domains. In the current section, the aims and objectives of the project are being discussed in relation to the results. The data resulted from the research is being interpreted in order to answer more research questions and explain why different models operate in particular ways.

7.1 In-domain evaluation

The first objective of this project was to create a learning framework that solves opinion mining learning problems. The whole process of learning and evaluating has been extensively described. Every part of this process has been chosen carefully according to the different requirements of each problem and a common framework for learning and evaluating has been developed (Section 5).

The second objective aimed in solving three general learning problems. Starting from binary classification and later with multi-class classification, regression and ranking numerous models were built and evaluated. Some of them were successful, even though some other failed to learn the problem. However, among various learning problems, the research was focused on those models that perform the best. As mentioned before, the aim of this research was not only to solve problems but also to move a step forward and deal with further challenges arising from those problems. More specifically, features that reveal opinion were also investigated and a comparison of the three learning problems was developed, in order to understand which the best model per learning problem is.

Features that reveal opinions depend on two factors, the learning algorithm and the conference. During the first experiments I tried to find patterns that are known by the bibliography to capture opinions like adjectives and POS combinations. However, using the whole review annotated by the POS or Stanford dependencies revealed more information than the isolated patterns. For binary classification algorithms trained on KDD appear to perform better for pairs of words rather than isolated words. Comparison among simple bigrams, bigrams tagged with POS, bigrams for which valence shifters have been extracted and finally Stanford dependencies did not reveal any statistically significant difference but it offered more information compared to isolated words like simple unigrams, unigrams tagged with POS and so on. In contrast, algorithms trained on ECML, that was predicted to be an easier conference to be learned, performed well independent of the feature set and both isolated features and pairs of features and offered the same amount of information. For regression and ranking I trained models using simple unigrams and bigrams, as generally they turned out to be the most effective. For this problem unigrams slightly outperformed bigrams. The process of extracting features that reveal opinions from the particular dataset became even more difficult due to the lack of opinion related words. The reviewers of research papers tend to be objective and not to use words that reveal their sentiment, they more often only use arguments that show why they want to accept or reject a paper. This fact explains why using simple unigrams or bigrams give adequate information about the polarity of the review.

For multi-class classification some of the models failed to predict specific classes. More specifically, all models failed to learn the three-class classification problem when trained on reviews from ECML. Confusion matrices showed that the problem is mainly between the adjacent classes. The most of the predictions converge to the mean scores. Two assumptions have been made here regarding the algorithms:

1) The learning algorithms are affected by the majority class. This is known in the bibliography as the imbalanced dataset problem, for which various methods have been developed in order to solve it (Monard & Batista 2002).
2) The learning algorithms are confused by the way the reviewers review research papers. For example, they use negative comments even if they accept a paper and positive comments even if they reject a paper.

The second assumption could explain why some models fail while others do not. For example, the fact that algorithms trained on ECML always fail to classify the reviews with scores belonging to the strong accept category while, algorithms trained on KDD do not fail could mean that the reviewers in the first case are more reserved to show their enthusiasm for papers they accept compared to the latter conference. This is in accordance with some first statistical results regarding the polarity of every score as retrieved by SentiWordNet Lexicon (Section 6.2.1). Moreover, it should be noticed that models trained on ECML successfully classify the reviews from KDD even if they fail within their domain. However, further investigation is needed in order to fully understand the particular process.

To sum up, in-domain evaluation shows that binary classification is the easiest problem to be learned based on the accuracy. Predicting review scores is more meaningful if we use regression instead of multi-class classification. Ranking is possible and independent of the score range. Reviewers use many neutral words which lead to a convergence of predictions around the mean scores. Learning each of these problems is possible but the performance is lower compared to other datasets like movie or product reviews as presented in the bibliography, which shows how complicated to analyze reviews of research papers is. ECML is an easier conference to be learned compared to KDD for all of the learning problems, which gives evidence that smaller score ranges are more meaningful.

7.2 Out-of domain evaluation, transfer learning, conference comparison

The second aim of this project was to compare the two conferences. Models trained on KDD are applied to ECML and vice versa. The learning framework has been extended in order to be used for applying transfer learning. The main challenge here was the different score rating of the two conferences. As a result, I had to investigate which scores of KDD correspond to scores of ECML based on the review text. Classification and regression cannot work with different score ranges between the two domains thus appropriate merging was required. In section 6.2.2 some early experiments using MNB show the similarity between reviews with scores 1, 2 and D and also the similarity between reviews with scores 5, 6 and A. Reviews corresponding to papers that have strongly been accepted and strongly been rejected, in other words with scores 1 - D and 6 – A respectively, are not so similar. These experiments revealed that merging reviews with specific scores is not a straight forward process.

Results from the findings reveal that, ranking is a more sensible problem to be transferred compared to classification and regression, an expected result of the current research. These results can be explained from the fact that ranking is independent of the score range of the two conferences. When trained using unigrams, SMOreg appeared to be not only a good regression model but also a good ranker both to rank reviews within the same domain as well as across domains. Classification and regression are possible only when the two conferences have the same score range. Regarding the difficulty of each learning problem similar conclusions with the in-domain evaluation have been reached. If one wants to separate the reviews into categories then binary classification is an easier problem to be transferred compared to multi-class classification based on the accuracy that models could reach. Multi-class classification predicted to be a very difficult task with the majority of models failing not only to be transferred but also to be learned. On contrast, score prediction is an easier problem if it is treated as a regression problem instead of multi-class classification, as the models always outperform the baseline and many of them are equivalent with the in-domain models. AR and SMO trained using simple unigrams turned out to be very promising schemes achieving low MAE when they are applied to different domains. Furthermore, the use of AR by the Sentiment analysis field is still limited, however, its use could be robust for different corpora as well.

Features that reveal opinions across domains depend on the learning algorithm, the learning problem and the direction that transfer learning is applied. As a result we cannot say that a specific feature set is the best for applying transfer learning. The fact that there are models that can be transferred shows that the two conferences do have similarities. An interesting question arising from this result, is why some models succeed and why other fail. In order to answer this question I compare the parameters of the same models when they are trained on different conferences. Table 26 shows eight models chosen from different learning problems with different performance when they are applied to a different domain. Each of these models has been trained and evaluated on KDD (model 1) and also trained and evaluated on ECML (model 2). Correlation corresponds to the correlation coefficient between the parameters of these two models (between model 1 and model 2).

For MNB I compare the likelihood estimates of the common features as calculated by models trained on ECML and KDD while for SMO I compare the weights that the two models give to the common features. I will start the analysis based on this table (Table 26) and on results from section 6.3. M1 and m2 are two models trained in order to learn the binary classification problem. M1 was the best model to be transferred during this task and the high correlation coefficient between the parameters of the models, trained from ECML and KDD respectively, shows why this model could be transferred. More specifically, high correlation shows that there is a linear dependence between the common features, in other words the probability of features given each class is similar for both conferences. Common features given the negative class have higher correlation to the common features given the positive class (correlation coefficient is equal to 0.96 and 0.92 respectively). M2 is a SVM model that could not been transferred neither from KDD to ECML nor from ECML to KDD. The low correlation coefficient shows that this model assigned different weights to the common features based on the conference that was used to train the algorithm. However, SVM could successfully apply to a different domain for *three-score regression* and could be applied from KDD to ECML for *four-score regression*. This is why models m6 and m7 present an increase to the correlation coefficient.

Interesting is the fact that even for models that failed to be applied to a different domain like m4 and m5 or they only presented one way transfer like m3 the correlation coefficient remains high. An explanation to this is that the common features are used with the same way by the reviewers from both conferences, but domain specific features that are not common are more discriminative within each conference. A final interesting observation is that model m1 has the lowest overlap of features between the two conferences but it was able to achieve the best performance and be applied to both directions. This shows that the common features are the most discriminative for both conferences.

Table 26 Correlation of parameters between different models.

Model id	Model	Learning problem	Performance when transferred	Correlation	Vocabulary overlap (%)
m1	MNB - SDep	Binary classification	successful	0.96/0.92	39.64
m2	SMO - SDep	Binary classification	unsuccessful	0.10	39.64
m3	MNB - bigramsPOS	Binary classification	one way transfer	0.98/0.96	62.29
m4	MNB - Unigrams	Four class classification	unsuccessful	0.99/0.99/0.99/0.97	65.52
m5	SMO – Bigrams	Four class classification	unsuccessful	0.04	49.51
m6	SMOreg - unigrams	3-score range regression	successful	0.15	66.16
m7	SMOreg - Unigrams	4-score range regression	one way transfer	0.23	65.52
m8	SMOreg - Unigrams	full-score range regression	unsuccessful	0.21	66.00

The last objective of this project was to search for alternative techniques in order to achieve a better performance. Frequently co-occurring entropy was the measure used in order to extract generalized features. Customizing MNB and SMO with different cutoffs of words with high FCE, led to improved performance for the models that could not be transferred during the binary classification task. Future work could include approaches that also use unlabeled data from the target domain during the training phase like those proposed by Tan et al. (2009) and Blitzer et al. (2007). Both papers achieved better results by using the knowledge of the target domain. Similar to the previous analysis I calculated the correlation coefficient between the models for different fractions of FCE words Figure 15. We observe that for MNB correlation is always very high, while for SMO correlation increases as far as a model can transfer the problem in a better way.

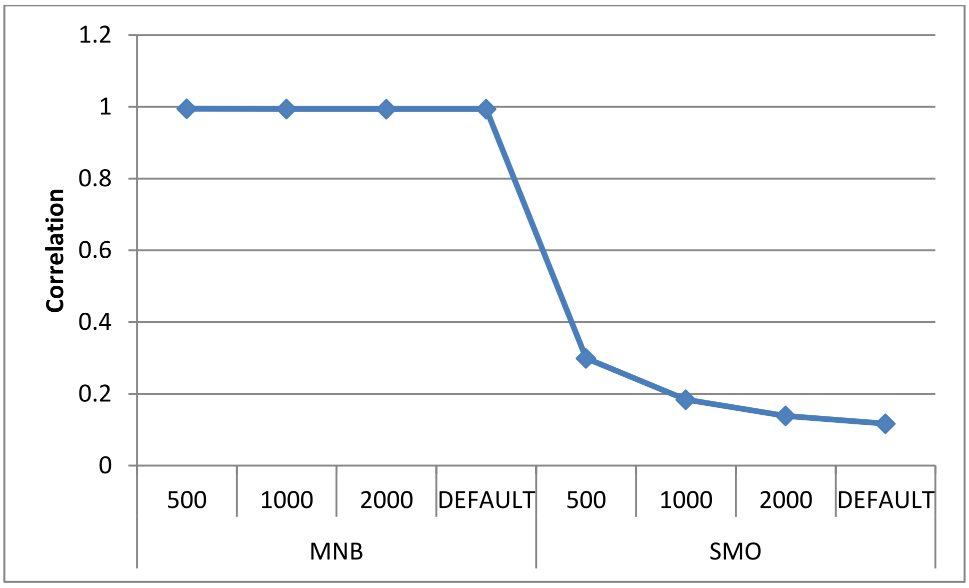


Figure 15 Correlation coefficient for different fractions of features with high FCE.

Many models have been trained on reviews from KDD, applied to reviews from ECML and vice versa. Generally, if we want to apply machine learning models across different domains then almost always we have to scarify the performance either in source or in target domain. Similar to other researchers, I assumed that the distribution of the features in the second conference given the class attribute is similar to the distribution of the features in the first conference given the class attribute (Aue & Gamon 2005, Blitzer et al. 2007). For some combinations of learning algorithms and feature sets this assumption holds while for other it does not, this has been interpreted as similarities and differences between the two conferences. The tables and figures below summarize the most interesting of them.

Table 27 Statistical similarities and differences (Section 6.2.1)

Score rating	KDD = 6 score range ECML = 4 score range
Review Length	Reviews from KDD have smaller length than reviews from ECML (195.80±30.11 versus 251.94±46.78)
Parts Of Speech	The same fraction of JJ, NN, RB, VB for both conferences. The higher the score the higher the fraction of adjectives for both conferences.
Polarity	The smaller the score the lower the polarity as estimated using SentiWordNet lexicon.
Valence shifters	Reviewers use overstatements more frequently than understatements for both conferences.

Table 28 The common words between the most similar reviews for KDD and ECML that express the most positive opinion as ranked using SMOreg – Unigrams ranker. KDDtoECML corresponds to the reviews, which were first or last in the rank when the ranker is trained and tested on KDD and when the same ranker is tested on ECML. ECMLtoKDD correspond to the reviews which were first or last in the rank when the ranker is trained and tested on KDD and when the same ranker is tested on ECML (ID: for In Domain evaluation, TL: for Transfer Learning or out-of domain evaluation, Score: the ground truth score).

Direction:ID→TL	**Rank**	**Score ID -TL**	**Common words between these reviews**
ECMLtoKDD	First	A - 5	for,of,a,i,method,to,would,and,have,paper,adapt,with,it,problem,the,in,term,an,approach,improv
ECMLtoKDD	Last	D - 3	need,for,of,on,limit,onli,-,ha,not,thei,-rrb-,?,data,-lrb-,do,a,us,should,i,what,to,but,been,experiment,such,and,compar,TAGNUMBER,valu,with,it,at,ar,the,in,veri,thi
KDDtoECML	First	2 - C	for,set,show,algorithm,of,could,-lrb-,find,with,it,a,the,and,thi,exampl,-rrb-,approach,propos,go
KDDtoECML	Last	2 - D	particular,these,ha,-,-rrb-,hard,a,i,what,to,emploi,requir,be,behind,learn,and,that,author,ar,veri,there,claim,why,for,no,languag,of,on,or,onli,not,thei,then,algorithm,clear,-lrb-,accuraci,should,experi,exampl,method,previou,but,function,howev,formal,have,state,TAGNUMBER,it,the,in,thi,their,also

Table 29 Similarities and differences based on the performance of machine learning models.

PROBLEM	DESCRIPTION
The most similar scores (Section 6.2.2, Table 9)	Reviews with scores 1 and 2 correspond to the reviews with score D Reviews with scores 5 and 6 correspond to the reviews with score A
Not similar scores (Section 6.2.2, Table 9)	Reviews with score 1 are not similar to the reviewers with score D. Reviews with score 6 are not similar to the reviewers with score A
Which learning problem is easier?	Binary classification is easier than multi-class classification. Regression more sensible for review score prediction. Ranking the only problem that can be learned and transfer independent the score range.
Which conference can easier be learned?	ECML is easier than KDD based on the performance of the models per task: Binary Classification Best model for KDD: 67.99±3.05% (MNB-Sdep) Binary Classification Best model for ECML: 69.45±4.35% (MNB-Sdep) 4 class Classification Best model for KDD: 40.63±2.76 (MNB-Unigrams) 4 class Classification Best model for ECML: 46.43±4.32% (SMO-BigramsPOS) Regression Best model for KDD: 1.3515±0.05 (SMOrbf-Unigrams) Regression Best model for ECML: 0.8328±0.03 (SMOrbf-Unigrams) Ranking Best model for KDD: 0.3636±0.03 (SMOreg-Unigrams) Ranking Best model for ECML: 0.2922±0.02 (SMOreg-Unigrams)
Binary classification (Section 6.3.1, Table 11)	MNB – Sdep the best performance within KDD and ECML, the only model that could be transferred in both directions. MNB trained using any other feature set shows that common features are used with a similar way (high correlation between the likelihood of common features) but other domain specific features are more discriminative for every conference (transfer learning fails). SMO always fail when transferred to different domains independent of the feature set.
Multi-class classification (Section 6.3.1, Table 12)	Learning the 3-class classification completely fails for ECML and transfer learning fails from KDD to ECML. Probably reviewers in ECML do not express their enthusiasm for the papers they accept with the same manner that reviewers in KDD do. 4-class classification is possible only within a domain but fails across domains. Transfer learning is possible only if we have the same number of classes.
Regression (Section 0, Table 22)	Transfer learning is possible only if we have the same score range. AR – 1000Unigrams the best model for applying knowledge to a different domain.
Ranking (Section 0, Table 25)	The only problem that can be transferred between the two conferences independent of the score range. SMOreg – Unigrams the best ranker.
The best models that can be transferred both from KDDtoECML and from ECMLtoKDD	Binary Classification: MNB-Sdep Regression: AR – 1000Unigrams Ranking: SMOreg-Unigrams

The tables below present opinion features that are more likely to present negative and positive sentiment as extracted by MNB-Sdep during the binary classification problem. APPENDIX G presents many more phrases that reveal opinions as extracted by different models. In addition to that, it summarizes the best model per learning problem and conference. Features have been ordered based on the ratio:

$$\text{score} = \frac{f(Neg, feature)}{f(Pos, feature)}$$

Where $f(Neg, feature)$ ($f(Pos, feature)$) corresponds to the likelihood of this feature to predict negative (positive) review as calculated by MNB.

Table 30 Top 10 Stanford dependencies with high probability of predicting negative as estimated by MNB – Sdep (Binary classification problem)

KDDtoKDD	**ECMLtoECML**
root(ROOT na)	amod(work existing)
root(ROOT kdd)	nsubj(is way)
nsubj(is contribution)	root(ROOT questionable)
dobj(see points)	cop(questionable is)
nsubj(lacks paper)	det(contribution no)
num(TAGNUMBER.TAGNUMBER TAGNUMBER.TAGNUMBER)	nn(set training)
aux(introduce to)	cop(significant is)
nsubj(lacks it)	aux(compare does)
root(ROOT lacks)	advmod(limited very)
aux(given have)	det(english the)

Table 31 Top 10 Stanford Dependencies with high probability of predicting positive based on MNB – Sdep (Binary classification problem).

KDDtoKDD	**ECMLtoECML**
xcomp(ienjoyed reading)	amod(topic important)
advmod(paper overall)	det(type a)
conj_and(strong weak)	amod(measure new)
dobj(like paper)	auxpass(formulated is)
dobj(addressing problem)	amod(paper solid)
root(ROOT overall)	advmod(presented clearly)
amod(paper interesting)	root(ROOT of)
nn(approach mining)	root(ROOT ienjoyed)
amod(approach interesting)	nn(paper conference)
aux(written could)	amod(paper good)

7.3 Performance

Finally, this section discusses the time and space requirements regarding the training of different models. These numbers do not include the time required for the parsing of each review and they only refer to the training of models. As a result, the time and space requirements for MNB and SMO are linear in the number of features. The maximum time needed to train the particular classifiers was two minutes. In contrast, SMO for regression converge a little slower than the classifiers and the training time depends on the choice of the complexity parameter as a higher value penalizes more attributes and results to faster training (approximately 5 to 10 minutes were enough). AR and Bag require many iterations until they coverage which resulted to around 15-20 minutes of training. Finally, RN is the slower model, since the use of neural networks need many iterations in order to find appropriate weights for the neurons and converge to the appropriate values, a process which resulted to several hours of training. The experiments have been performed on a laptop with characteristics as summarized in the table below.

Manufacturer	Samsung Electronics
Operating system	Windows 7 Home Premium (64 - Bit)
Processor	Inter(R) Core(TM) i7-2670QM CPU @ 2.20GHz
RAM	4.00GB

8 Conclusions and Future Work

This project was developed using supervised machine learning techniques in order to extract opinions and compare reviews of research papers submitted to two major international conferences KDD - 2009 and ECML PKDD – 2012. Machine learning algorithms have been trained using review texts and review scores in order to build models that solve three general learning problems: Classification in order to predict the category in which the reviewer will place a paper, based on the opinions he expresses on the review text. Regression, which is an alternative way of predicting numerical scores or in this case review score ratings. Finally, ranking in order to rank the reviews from those that express the most positive opinion to those that express the most negative opinion. Models are applied from one conference to the other in order to transfer learning and many similarities and differences have been revealed based on their performance.

Many different learning problems have been solved, transferred and compared, even though most of the researchers tend to investigate them separately. More specifically, this research explained which learning problem can easier be solved and transferred, what are the best models for doing that and what alternative ways exist. The way that reviewers express their opinions have been analyzed in order to answer research questions like: how difficult is to predict a reviewer's score? Is it easier to learn preferences? At the same time, this study is one of the first to compare reviews from different conferences using machine learning techniques. Many experiments extract frequent opinion features between the two conferences, similarities and differences, models that can be transferred and models that cannot including the reason of failure or success.

The proposed learning framework has been instantiated with state-of-the-art machine learning algorithms. The good reputation of MNB and SVM is also supported by the outcomes of this work. The two classifiers present equal performance when they are evaluated within a specific domain. However, while, SVM are good models within a particular domain they perform poor when they are applied to a different domain. Moreover, SMOreg turned out to be a good model for regression and ranking both within and across domains. Finally, Additive Regression using Model trees achieves the best overall performance, which is also recommended as a good model for applying transfer learning. While sentiment classification and regression have extensively been explored by other researchers in *Opinion mining* area, ranking is not a famous domain of study. In the current work, the ranking problem has also been investigated and it has been useful for learning and comparing the two conferences.

Although the aims and objectives of this project have been completed, there is always room for improvement. Reviewers of research papers also comment on strong and weak points of each paper, fields that could give even more information regarding the similarities and differences between the two conferences. Not only that, they could also improve the performance of the learning schemes. Advanced feature extraction approaches like the combination of many feature sets as proposed by Ng et al. (2006) could also improve the performance. The proposed method reaches high accuracy by using unigrams, bigrams and trigrams, polarity words and syntactic dependency relations, in combination with a feature space reduction. Their experiments indicate that using each of the feature sets on each own sometimes increase and sometimes decreases the accuracy. However, a combination of those is able to reach the best possible results. Based on that, more investigation regarding the feature combination could be performed. RankNet, which is a neural network ranking model, performed very poor, however, many other rankers exist, that one could experiment. Finally, *Transductive transfer* learning or *inductive transfer* methods from *transfer learning* area could probably transfer the learning better compared to traditional machine learning approaches.

9 Bibliography

Aue, A. & Gamon, M., 2005. Customizing Sentiment Classifiers to New Domains : a Case Study. In *Proceedings of RANLP*.

Baccianella, S., Esuli, A. & Sebastiani, F., 2009a. Evaluation Measures for Ordinal Regression. In *Ninth International Conference on Intelligent Systems Design and Applications*. Ieee, pp. 283–287.

Baccianella, S., Esuli, A. & Sebastiani, F., 2009b. Multi-facet rating of product reviews. *Advances in Information Retrieval*, 5478(9), pp.461–472.

Blitzer, J., Dredze, M. & Perera, F., 2007. Biographies , Bollywood , Boom-boxes and Blenders : Domain Adaptation for Sentiment Classification. In *Association for Computational Linguistics*. pp. 440–447.

Breiman, L., 1996. Bagging predictors. *Machine Learning*, 24(2), pp.123–140.

Burges, C. et al., 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine learning*. Bonn, Germany: ACM Press.

Chawla, N.V., Japkowicz, N. & Drive, P., 2004. Editorial : Special Issue on Learning from Imbalanced Data Sets Aleksander Ko l cz. , 6(1), pp.2000–2004.

Dasgupta, S. & Ng, V., 2009. Mine the easy, classify the hard: a semi-supervised approach to automatic sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Singapore: Association for Computational Linguistics, pp. 701–707.

Dave, K., Lawrence, S. & Pennock, D.M., 2003. Mining the Peanut Gallery : Opinion Extraction and Semantic Classification of Product Reviews. *Word Journal Of The International Linguistic Association, ACM*, 17, pp.519–528.

Demsar, J., 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning research*, 7, pp.1–30.

Dietterich, T., 2000. Ensemble methods in machine learning. In *Proceedings of the 1st International Workshop in Multiple Classifier Systems*. pp. 1–15.

Duh, K., 2008. Ranking vs. regression in machine translation evaluation. In *Proceedings of the Third Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pp. 191–194.

Esuli, A. & Sebastiani, F., 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of International Conference on Language Resources and Evaluation (LREC)*. Genoa, pp. 417–422.

Fan, Wen, Sun, S. & Song, G., 2011. Probability Adjustment Naïve Bayes Algorithm Based on Nondomain-Specific Sentiment and Evaluation Word for Domain-Transfer Sentiment Analysis. In *Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. IEEE, pp. 1043–1046.

Flach, P., *Machine learning experiments, to appear in the forthcoming Encyclopedia of Machine Learning (Springer),*

Fürnkranz, J. & Hullermeier, E., 2011. *Preference Learning* First ed., Berlin: Springer.

Graf, A. & Borer, S., 2001. Normalization in support vector machines. *Pattern Recognition*.

Hsu, C., Chang, C. & Lin, C., 2003. A practical guide to support vector classification. , pp.1–12.

Japkowicz, N. & Shah, M., 2011. *Evaluating Learning Algorithms: A Classification Perspective*, New York, NY, USA: Cambridge University Press.

Kennedy, A.K. & Inkpen, D.I., 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2).

Kim, S. et al., 2006. Automatically Assessing Review Helpfulness. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP ’06)*. pp. 423–430.

Lachiche, N. & Flach, P., 2003. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. In *ICML*.

Lin, Z., Tan, S. & Cheng, X., 2011. Using Key Sentence to Improve Sentiment Classification. In *Information Retrieval Technology*. Springer-Verlag Berlin, pp. 422–433.

Liu, B., 2010. Sentiment analysis and subjectivity. *Handbook of Natural Language Processing,*, pp.1–38.

Marneffe, M.D., MacCartney, B. & Manning, C., 2006. Generating typed dependency parses from phrase structure parses. *Proceedings of LREC*.

Mingzhi, C. et al., 2009. A Random Walk Method for Sentiment Classification. In *Second International Conference on Future Information Technology and Management Engineering*. Ieee, pp. 327–330.

Monard, M. & Batista, G., 2002. Learning with skewed class distributions. *Advances in Logic, Artificial Intelligence and its Applications*, pp.1–9.

Ng, V., Dasgupta, S. & Arifin, S.M.N., 2006. Examining the Role of Linguistic Knowledge Sources in the Automatic Identification and Classification of Reviews. In *Proceedings of the COLINGACL on Main conference poster sessions*. Association for Computational Linguistics.

Ohana, B. & Tierney, B., 2009. Sentiment classification of reviews using SentiWordNet. In *9th. IT & T Conference*. Dublin.

Pan, S. & Yang, Q., 2010. A survey on transfer learning. *Knowledge and Data Engineering, IEEE*, 22(10).

Pang, B. & Lee, L., 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the Association for Computational Linguistics*. Cornell University Library, pp. 271–278.

Pang, B. & Lee, L., 2006a. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 1(2), pp.91–231.

Pang, B. & Lee, L., 2006b. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Pang, B., Lee, L. & Vaithyanathan, S., 2002. Thumbs up ? Sentiment Classification using Machine Learning Techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pp. 79–86.

Raina, R., Ng, A. & Koller, D., 2006. Constructing informative priors using transfer learning. In *Proceedings of the 23rd international conference on Machine learning ICML 06*. ACM Press, pp. 713–720.

Stone, P., Dunphy, D. & Smith, M., 1966. The General Inquirer: A Computer Approach to Content Analysis. *MIT Press, Cambridge, Massachusetts*.

Tan, S. et al., 2009. Adapting naive Bayes to domain adaptation for sentiment analysis. In *Advances in Information Retrieval*. SPRINGER-VERLAG BERLIN, pp. 337–349.

Tang, H., Tan, S. & Cheng, X., 2009. A survey on sentiment detection of reviews. *Expert Systems with Applications*, 36(7), pp.10760–10773.

Toutanova, K. et al., 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology NAACL 03*. pp. 173–180.

Toutanova, K. & Manning, C.D., 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*. pp. 63–70.

Turney, P., 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. Philadelphia, pp. 417–424.

Wang, H. et al., 2003. Mining concept-drifting data streams using ensemble classifiers. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’03*, p.226.

Witten, I. & Frank, E., 2005. *Data Mining Practical Machine Learning Tools and Techniques* Second edi., Morgan Kaufmann.

Zhang, Z. & Varadarajan, B., 2006. Utility scoring of product reviews. In *Proceedings of the 15th ACM international conference on Information and knowledge management CIKM 06*. ACM press, pp. 51–57.

Zhou, L. & Chaovalit, P., 2005. Movie Review Mining : a Comparison between Supervised and Unsupervised. In *Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS)*. IEEE, pp. 1–9.

APPENDIX A Machine learning algorithms

Let $\{\vec{x_i}, y_i\}$, $i = 1 \ldots m$ denote a set of m training examples, where each input vector $\vec{x_i}$ is a different review which consists of k features $\vec{x_i} = \{x_{i1}, x_{i2}, \ldots x_{ik}\}$ while $y_i \in \{c_1, .., c_n\}$ is the class label. x, is a new test instance with unknown class label C.

Naïve Bayes and Multinomial Naïve Bayes

Naïve Bayes is a probabilistic classifier which has been widely used in many different fields of machine learning for solving real-world problems such as spam filtering, document retrieval and in our case sentiment classification. It is based on Bayes theorem but simplifies it by assuming independence among the inputs. Inputs here are the different attributes or features. This assumption is made in order to simplify the difficult calculations that Bayesian learning requires. After a Naïve Bayes classifier is trained it can describe the probability of any new test instance to belong in every possible class according to the classification task. This can be defined as:

$$P(c_j|x) = P(c_j) * \prod_i P(a_i|c_j)$$

$$where\ c_j = c_{1..n}\ and\ a_i\ the\ attribute\ in\ position\ i\ of\ the\ test\ instance\ x$$

So during the training phase the probability of every attribute belonging to a class cj has to been calculated for all of the classes according to:

$$P(a_i|c_j) = \frac{n_{cjai} + 1}{n_{cj} + m}$$

Where n_{cjai} is the number of training examples with class c_j and attribute a_i and n_{cj} is the number of training examples with class c_j. m is the number of possible values of attribute a_i. This term is used in order to avoid zeros and is known as the Laplace correction. Note here that when Naïve Bayes is used in text documents or when the attributes are represented by their frequence (also known as Multinomial Naïve Bayes) then n_{cjai} is the number of times attribute a_i occurred in documents of class c_j and n_{cj} is the total number of words in documents of class c_j. The term m is replaced by the |Vocabulary| which is the number of all distinct words in all of the documents in the training set. Finally if an attribute corresponds to continuous values for example an attribute that represents the age. Then it can be approximated by a probabilistic function. In the most of the probability density of Gaussian distribution is used, so, the aim is to calculate the mean and standard deviation for this attribute.

The priors have also to been calculated and they are simply the probability of every class defined as: the number of training examples belonging to class c_j. divided by the total number of training examples: $P(c_j) = \frac{number\ of\ training\ examples\ of\ class}{total\ number\ of\ training\ examples}$

During the test phase. A new unlabeled test instance is preprocessed in order to contain only attributes that have been appeared during the training phase or for text documents, terms that belong to the Vocabulary. The final class that Naïve Bayes will assign to the test instance (or document) x is simple the one that has the highest value according to:

$$\underset{c_j}{\operatorname{argmax}} P(c_j|x), \qquad j = 1..n$$

As we said in the beginning this classifier makes the assumption that the probability of one attribute appearing does not change the probabilities for any other attribute to appear after this. Of course this does not always hold especially in text documents where attributes are words however this classifier still works by achieving high accuracy.

Support Vector Machines

Support Vector Machines (SVMs) are a group of supervised learning algorithms that can be applied for classification and regression. In this section we will describe how classification is possible. Information in this section is based on Witten and Frank (2005: 215 - 219).

Support Vector machines search for the *maximum margin hyperplane*. Hyperplane is a different word for a linear model. Figure 16 illustrates a simplify case where a SVM finds the maximum margin hyperplane. Suppose we have a binary classification problem where the two classes are linearly separated. Then the SVM will find the hyperplane (line) that classifies correct all of the training examples and also this will be the maximum margin hyperplane which means that it will be in the maximum possible distance for both classes or in other words it will not come closer to the data of any of the two classes. Support vectors are the instances that are closer to the maximum margin hyperplane. Once the support vectors are defined for the two classes then all of the other instances can be omitted. The basic assumption of SVMs is that a general model which will perform well in future data is the one that separates the training data as far as possible. The relation that describes the hyperplane and can be used in order to classify new test instances can be defined according to:

$$y = b + \sum_{i \text{ is a Support Vector}} a_i y_i \overrightarrow{s(\iota)} \cdot \vec{x}$$

Where a_i and b are numeric parameters that the algorithm has to determine, $\vec{x}$ is a vector that represents a test instance $\overrightarrow{s(\iota)}$is a vector that represents one of the support vectors y_i is the class value for this support vector and $\overrightarrow{s(\iota)} \cdot \vec{x}$ is the dot product of the test instance with the support vector i.

An extension of linear SVM which is able to create nonlinear class boundaries by using linear models has also been proposed. In order to find a good hyperplane SVMs transform the training data into a new space using a nonlinear mapping (Figure 16). Basic component in this approach is the kernel function that the SVM uses in order to calculate the dot product between the support vector and the test instance. The representation of the instances in a new feature space leads to transformed high dimensional data and kernel function is essential in order to perform the required calculations. The linear SVM algorithm now operates in the new feature space and searches there for the maximum margin hyperplane, this hyperplane possibly will not be linear for the original feature space so the algorithm has created a nonlinear class boundary.

SVMs perform very well for many real-problems. The fact that they search for a large margin makes them to create very stable models that perform well for new data. Moreover, it is very difficult to overfit the training data. Disadvantage is that algorithms of this group have higher computational complexity compared to other classifiers especially the non linear approach. However, fast implementations have been proposed that reduce the complexity.

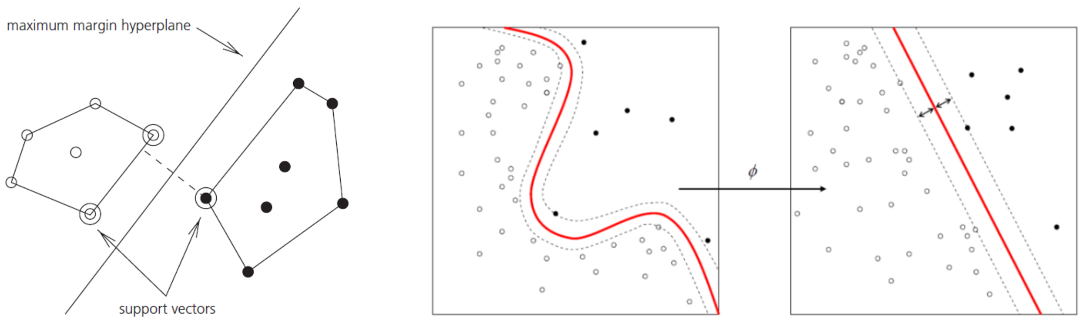


Figure 16 The figure in the left (taken from Witten & Frank, (2005: 216)) shows the maximum margin hyperplane and support vector machines for a binary classification task where the filled and open circles correspond to the two classes. The hyperplane is in the middle and orthogonal to the dashed line which indicates the minimum distance of the two classes. The figure in the left (taken from [8]) shows the transformation of the original feature space(left) to a new feature space (right) where linear separation is performed by a SVM.

Additive Regression

Ensemble methods (Dietterich 2000) combine many models, known as experts, in order to take the final decision. They have been a very popular research topic during the last decade due to the fact that they offer an appealing solution to several interesting learning problems such as improving prediction accuracy, scaling inductive algorithms to large databases and learning from concept-drifting data streams (H. Wang et al. 2003). Only a few papers (Z. Lin et al. 2011) use ensembles of classifiers in Sentiment Analysis field. Both additive Regression and Bagging (described next) are two methods that combine different models in order to predict continuous numeric values.

Additive regression starts with a simple regression model which is trained and evaluated. The instances that this model predicts incorrect are used in order to train a second regression model. More specifically, the difference between the ground truth value and the predicted one is calculated. These differences are called *residuals* and are used as the new ground truth label. The second model is fitted with the new instances and the whole process is repeated until a measure of performance like the root mean squared error is minimum. For this project Model trees are used as the base models which is the implementation that Weka suggests (Witten & Frank 2005: p. 325 - 327).

Bagging

Similar to additive regression Bagging (Bootstrap AGGregatING) is a different approach, which has been proposed by Breiman (1996), in order to combine the decisions of different models and can be used both for classification and for regression. In this project it has been adopted for numeric prediction. The way it combines the decision of the different regression models is very simple. A homogeneous set of regression models is trained with different subsets of the training instances. For a new test instance the predictions of all models are taken into consideration and the average of them is the final decision. For this project, similar to Additive Regression, the ensemble consists of Model trees.

RankNet

RankNet is a pairwise ranking methods. The general concept in pairwise ranking methods is to take pairs of documents and decide which of the two should be preferred. This implies a binary classification problem. It is obvious that after a ranker is trained if a new unlabeled document has to be rank then the error will always be equal to 0. Pairwise ranking methods

[8] http://en.wikipedia.org/wiki/File:Kernel_Machine.png

are meaningful only in cases that a set of at least 2 unlabeled documents have to be shorted. RankNet is a neural network ranking model proposed by Burges et al. (2005) and used for document retrieval by search engines. This is a probabilistic method which uses neural networks as models and gradient descent as the algorithm in order to optimise the loss function. During the training phase a probability is assigned to every pair of documents $\vec{x_i}, \vec{x_j}$, where $i \neq j$ according to (1).

$$\bar{P}_{\vec{x_i},\vec{x_j}} = \begin{cases} 1\ if\ \vec{x_i} \succ \vec{x_j}\ or\ y_i > y_j \\ 0.5\ if\ \vec{x_i} = \vec{x_j}\ or\ y_i = y_j \\ 0\ \ otherwise \end{cases}$$

(1)

We denote with $\bar{P}$ the true probability for a pair of documents. This is the probability that $\vec{x_i}$ is higher in the rank than $\vec{x_j}$.

A neural network is used in order to model probabilities between every pair of documents during the training phase according to (2).

$$P_{\vec{x_i},\vec{x_j}} = \frac{\exp\left(f(\vec{x_i}) - f(\vec{x_j})\right)}{1 + \exp\left(f(\vec{x_i}) - f(\vec{x_j})\right)}$$

(2)

Where $f(\vec{x_i})$ is the score that maps the neural network to the document $\vec{x_i}$.

$P_{\vec{x_i},\vec{x_j}}$ is compared with the true $\bar{P}_{\vec{x_i},\vec{x_j}}$ and cross entropy is used as the loss function that gives a penalty for incorrect ranking (3). This loss function gives different penalties which are proportional to the importance of the error. For example, bigger penalty will be given if the model predicts that $f(\vec{x_1}) > f(\vec{x_2})$ when the true labels are $y_1 < y_2$ and smaller penalty if the model predicts that $f(\vec{x_1}) = f(\vec{x_2})$. The reader can refer to Burges, Renshaw, & Deeds, (1998) where an extensive proof for the properties of the loss function is given.

$$loss\left(P_{\vec{x_i},\vec{x_j}}, \bar{P}_{\vec{x_i},\vec{x_j}}\right) = -\bar{P}_{\vec{x_i},\vec{x_j}} \log\left(P_{\vec{x_i},\vec{x_j}}\right) - \left(1 - \bar{P}_{\vec{x_i},\vec{x_j}}\right) \log\left(1 - P_{\vec{x_i},\vec{x_j}}\right)$$

(3)

Finally, the aim is to optimise the cross-entropy loss. So, after the cost is calculated the model's parameters (weights in neural networks) are updated via stochastic gradient descent in order to reduce the cost.

I presented RankNet which is a commonly used pairwise ranking algorithm however, many more algorithms are available in the bibliography. RankLib[9] and SVM-RANK[10] are two open source libraries that have implementations of both pairwise and listwise ranking algorithms the first one has been used for the scope of this project.

[9] http://people.cs.umass.edu/~vdang/ranklib.html

[10] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

APPENDIX B Software and libraries

Here is the list of libraries, lexical resources and software that I have adopted for this project:

- Stanford parser (v. 2.0.2 /2012-05-22) and Stanford POS tagger (v 3.1.2 /2012-05-22): Stanford NLP group offers a fast Java package which is used to parse text documents. Stanford Log-linear Part-Of-Speech Tagger (Toutanova et al. 2003) is used in order to tag words with their Parts Of Speech. Stanford typed dependency parser (Marneffe et al. 2006) returns pairs of words that reveal typed dependencies based on grammatical relations like *subject* or *indirect object.*
- WEKA (v. 3.5.7) (Witten & Frank 2005) is a well-developed Java package that supports a collection of machine learning algorithms for data mining tasks. More specifically, it supports data pre-processing, classification, regression, clustering, association rules, and visualization.
- RankLib (v. 2.0) is an open source JAVA library, developed by Van Dang[11], that has implementations of both pairwise and listwise learning to rank algorithms. The library is available on: http://people.cs.umass.edu/~vdang/ranklib.html
- Snowball is a popular package to perform stemming process, which reduces inflected (or sometimes derived) words to their stem or base form. It is integrated with WEKA.
- SentiWordNet (v. 3.0) (Esuli & Sebastiani 2006) is an opinion lexicon derived from the WordNet database. It contains English terms where each of them is associated with numerical scores that correspond to positive and negative sentiment information. It is publicly available for research purposes.
- General Inquirer (Stone et al. 1966) is a dictionary that annotates English word senses. It includes tags that label them as positive, negative, overstatement understatement etc.
- Commons Math (v. 3.0)[12]: Commons Math is an open source JAVA library that contains statistics and mathematics components like classes for Pearson and Spearman’s correlation.

[11] PhD student supervised by Professor W. Bruce Croft at Center for Intelligent Information Retrieval (CIIR), Department of Computer Science, University of Massachusetts Amhers.
[12] Developers, C. M. (2010). Apache Commons Math, Release 3.0. The Apache Software Foundation. Retrieved from http://commons.apache.org/math/index.html

APPENDIX C Feature Sets

Table 32 and Table 33 summarize the feature sets that have been created during the experimental phase in section 6.

Table 32 Feature sets

Feature set	**Description**
bigramsPOS	Bigrams tagged with their Part Of Speech.
bigramsPOSVal	Bigrams tagged with their Part Of Speech and Valence shifters replaced by tags the OVS for overstatements and UND for understatements.
SDep	Stanford dependencies
unigramsPOS	Unigrams tagged with their Part Of Speech
unigramsPOSVal	Bigrams tagged with their Part Of Speech and Valence shifters replaced by tags the OVS for overstatements and UND for understatements.
Bigrams	Simple bigrams (pairs of sequential words)
Unigrams	Simple unigrams (isolated words)

Table 33 Patterns as extracted using the Stanford Log-linear Part-Of-Speech Tagger (Toutanova et al. 2003) and the Stanford typed dependency parser (Marneffe et al. 2006). The first is used in order to tag words with their Parts Of Speech. The latter returns pairs of words that reveal typed dependencies based on grammatical relations like *subject* or *indirect object*.

Parts Of Speech	**Description**	**Stanford Dependencies**	**Description**
JJ	Isolated Adjectives	MOD	Modifier
NN	Isolated Nouns	COMP	Complement
RB	Isolated Adverbs	SUBJ	Subject
VB	Isolated Verbs	NEG	Negation
JJ_NN	Sequential Adjective/Noun pairs		
NN_RB	Sequential Noun/Adverb pairs		
PRP_NN	Sequential Prp/Noun pairs		
RB_JJ	Sequential Adv/Adjective pairs		
RB_VB	Sequential Adverb/Verb pairs		
VB_NN	Sequential Verb/Noun pairs		
VB_RB	Sequential Verb/Adverb pairs		

Table 34 shows an example of a review when it is ran through different parsers. In the first case I use Stanford Part of Speech tagger in order to tag every word with its parts of speech and also General Inquirer lexicon in order to replace every word that is detected to be an overstatement or understatement with a single tag: OVS for overstatement and UND for understatement. In the second case the text is parsed using Stanford typed dependency parser. The parser returns pairs of words that reveal typed dependencies based on grammatical relations like *subject* or *indirect object*.

Table 34 Example of a review when it is annotated using different parsers.

Initial review	**"…Experiments are of very good quality and strongly supports the paper…".**
Review annotated with POS and valence shifters	"experi/NNS ar/VBP of/IN veri/RB OVS qualiti/NN and/CC strongli/RB support/VBZ the/DT paper/NN ./.""
Review is parsed using Stanford typed dependency parser.	"nsubj(are,experiments) nsubj(supports,experiments) root(ROOT,are) advmod(good,very) amod(quality,good) prep_of(are,quality) advmod(supports,strongly) conj_and(are,supports) det(paper,the) dobj(supports,paper)"

Multinomial Naïve Bayes has been trained in order to solve binary classification problems where reviews with scores 1, 2, 3 (D, C) have been treated as negative and reviews with score 4, 5, 6 (B, A) have been treated as positive. Tables bellow show the accuracy percentage and the average ranks (the last two columns) regarding the in-domain(ID) and out-of domain(TL) evaluation for all of the extracted patterns and all domain combinations.

Table 35 POS patterns

model	KDDto KDD	KDDto ECML	ECMLto ECML	ECMLto KDD	ID Average rank	TL Average rank
BigramsPOS	67.13%	67.82%	68.24%	64.76%	1.5	1.5
JJ_NN	64.02%	61.97%	61.75%	61.35%	12	11.5
NN_RB	64.33%	61.31%	60.95%	62.82%	12	11.5
PRP_NN	64.39%	62.05%	62.23%	64.52%	9	6.5
RB_JJ	64.76%	61.96%	61.91%	64.56%	9.5	7
RB_VB	64.83%	63.08%	62.72%	64.03%	6.5	5.5
VB_NN	65.7%	62.07%	62.15%	64.26%	6	7
VB_RB	63.53%	62.54%	62.07%	63.19%	11.5	7.5
UnigramsPOS	66.19%	64.37%	68.65%	65.18%	1.5	1.5
JJ	65.14%	62.69%	66.55%	64.45%	4	5
NN	64.95%	61.51%	63.11%	62.52%	5.5	11.5
RB	65.69%	63.38%	63.92%	63.81%	4	5.5
VB	64.45%	62.32%	62.47%	62.26%	8	9.5

Table 36 Stanford dependencies patterns

model	KDDtoKDD	KDDtoECML	ECMLtoECML	ECMLtoKDD	ID Average rank	TL Average rank
SDep	67.99%	66.54%	69.45%	66.77%	1	1
COMP	63.77%	60.75%	58.78%	62.34%	4.5	4.5
MOD	66.19%	64.51%	66.96%	64.03%	2.25	2.25
NEG	63.21%	60.36%	66.96%	64.03%	3.75	3.75
SUBJ	64.33%	63.59%	63.92%	62.96%	3.5	3.5

APPENDIX D Regression results

The Table in the right shows all of the competing models regarding the regression problem.

Model id	Model
m1	Baseline – mean score
m2	AR - 1000Bigrams
m3	AR - 1000Unigrams
m4	Bag - 1000Bigrams
m5	Bag - 1000Unigrams
m6	SMOreg - Bigrams
m7	SMOrbf - Bigrams
m8	SMOrbf - Unigrams
m9	SMOreg - Unigrams

Table 37, Table 38 and Table 39 show the mean absolute error and the rank of the competing models for all domain combinations regarding the 3-score, 4-score and full-score regression problem respectively. The models that could be applied to a different domain without having statistically significant difference with the corresponding in-domain models are underlined.

Table 37 three-score regression problem

model	KDDtoKDD	R	KDDtoECML	R	ECMLtoECML	R	ECMLtoKDD	R
m1	0.7289±0.0	9.0	0.7289±0.0	9.0	0.7215±0.0	9.0	0.7215±0.0	9.0
m2	0.6944±0.02	8.0	0.676±0.01	7.0	0.6722±0.04	8.0	0.6806±0.0	6.0
m3	0.6941±0.03	7.0	0.6544±0.01	1.0	0.6454±0.04	3.0	0.671±0.01	3.0
m4	0.6919±0.02	5.0	0.6739±0.0	6.0	0.6667±0.02	5.0	0.6725±0.0	4.0
m5	0.6937±0.02	6.0	0.6599±0.01	4.0	0.6695±0.01	7.0	0.6895±0.0	8.0
m6	0.676±0.01	2.0	0.6688±0.0	5.0	0.6411±0.02	2.0	0.6685±0.0	1.0
m7	0.6892±0.01	4.0	0.688±0.0	8.0	0.6676±0.01	6.0	0.6834±0.0	7.0
m8	0.678±0.02	3.0	0.6552±0.0	2.0	0.6392±0.02	1.0	0.6746±0.0	5.0
m9	0.6728±0.01	1.0	0.6577±0.0	3.0	0.6501±0.02	4.0	0.6709±0.0	2.0

Table 38 four-score regression problem

model	KDDtoKDD	R	KDDtoECML	R	ECMLtoECML	R	ECMLtoKDD	R
m1	1.0±0.0	9.0	1.0±0.0	9.0	1.0±0.0	9.0	1.0±0.0	9.0
m2	0.966±0.04	8.0	0.9448±0.01	7.0	0.9007±0.07	6.0	0.9456±0.01	8.0
m3	0.9325±0.04	2.0	0.8889±0.01	1.0	0.8489±0.06	3.0	0.9217±0.01	5.0
m4	0.9534±0.02	6.0	0.9266±0.01	6.0	0.9065±0.04	8.0	0.9267±0.01	6.0
m5	0.9382±0.03	3.0	0.8922±0.01	2.0	0.8669±0.02	5.0	0.9193±0.01	4.0
m6	0.9306±0.02	1.0	0.9234±0.0	5.0	0.8537±0.02	4.0	0.9145±0.0	1.0
m7	0.9481±0.02	5.0	0.9455±0.0	8.0	0.9021±0.03	7.0	0.9442±0.0	7.0
m8	0.941±0.03	4.0	0.8999±0.01	3.0	0.8328±0.03	1.0	0.917±0.0	3.0
m9	0.955±0.03	7.0	0.9209±0.01	4.0	0.8396±0.02	2.0	0.9166±0.0	2.0

Table 39 full-score regression problem

model	KDDtoKDD	R	KDDtoECML	R	ECMLtoECML	R	ECMLtoKDD	R
m1	1.5±0.0	9.0	1.0676±0.0	9.0	1.0±0.0	9.0	1.7604±0.0	9.0
m2	1.4175±0.1	8.0	1.0095±0.01	8.0	0.9007±0.07	6.0	1.637±0.02	2.0
m3	1.3251±0.06	1.0	0.9991±0.01	6.0	0.8489±0.06	3.0	1.6111±0.02	1.0
m4	1.4132±0.06	7.0	0.9896±0.0	4.0	0.9065±0.04	8.0	1.6467±0.01	4.0
m5	1.3711±0.03	3.0	0.9839±0.01	3.0	0.8669±0.02	5.0	1.6431±0.01	3.0
m6	1.3714±0.03	4.0	0.9965±0.0	5.0	0.8537±0.02	4.0	1.6516±0.0	7.0
m7	1.3952±0.02	6.0	1.0063±0.0	7.0	0.9021±0.03	7.0	1.7033±0.0	8.0
m8	1.3515±0.05	2.0	0.9706±0.01	2.0	0.8328±0.03	1.0	1.6504±0.0	5.0
m9	1.3717±0.05	5.0	0.9589±0.0	1.0	0.8396±0.02	2.0	1.6511±0.0	6.0

APPENDIX E Ranking results

The Table in the right shows all of the competing models regarding the ranking problem. Table 40, Table 41 and Table 42 show the rank error and the rank of the competing models for all domain combinations regarding the 3-score 4-score and full-score ranking problem respectively. The models that could be applied to a different domain without having statistically significant difference with the corresponding in-domain models are underlined.

Model id	Model
m1	AR - 1000Unigrams
m2	MNB - Unigrams
m3	RN - 1000Unigrams
m4	SMOreg - Unigrams

Table 40 three-score ranking problem

model	KDDtoKDD	R	KDDtoECML	R	ECMLtoECML	R	ECMLtoKDD	R
m1	0.3869±0.04	3.0	0.3484±0.02	3.0	0.3361±0.05	3.0	0.3699±0.02	2.0
m2	0.358±0.04	1.0	0.3322±0.0	2.0	0.3007±0.04	1.0	0.3847±0.01	3.0
m3	0.4004±0.03	4.0	0.369±0.01	4.0	0.341±0.03	4.0	0.4063±0.01	4.0
m4	0.3668±0.03	2.0	0.3207±0.01	1.0	0.317±0.04	2.0	0.3691±0.0	1.0

Table 41 four-score ranking problem

model	KDDtoKDD	R	KDDtoECML	R	ECMLtoECML	R	ECMLtoKDD	R
m1	0.3696±0.03	3.0	0.3494±0.01	3.0	0.3249±0.03	3.0	0.3734±0.01	2.0
m2	0.3694±0.03	2.0	0.3482±0.01	2.0	0.3158±0.02	2.0	0.3927±0.01	3.0
m3	0.4035±0.03	4.0	0.3789±0.02	4.0	0.3557±0.02	4.0	0.4102±0.01	4.0
m4	0.3624±0.03	1.0	0.3183±0.0	1.0	0.2922±0.02	1.0	0.366±0.0	1.0

Table 42 full-score ranking problem

model	KDDtoKDD	R	KDDtoECML	R	ECMLtoECML	R	ECMLtoKDD	R
m1	0.3666±0.02	2.0	0.3454±0.01	2.0	0.3249±0.03	3.0	0.3777±0.01	2.0
m2	0.3747±0.03	3.0	0.3482±0.01	3.0	0.3158±0.02	2.0	0.3949±0.01	3.0
m3	0.4045±0.04	4.0	0.3659±0.01	4.0	0.3557±0.02	4.0	0.4138±0.01	4.0
m4	0.3636±0.03	1.0	0.3234±0.0	1.0	0.2922±0.02	1.0	0.3694±0.0	1.0

APPENDIX F Statistical Tests

The tables below summarize all of the significance tests performed in order to find: A. if the best model per domain and learning problem is significantly better compared to the other models B. what are those models that can really been transferred across domains.

A. Significance test for the best model regarding the in-domain evaluation

Table 43 Classification (binary classification on the left, four-class classification on the right)

TEST (DOMAIN)	Paired t-test (KDD)	Paired t-test (ECML)
Best model →	MNBSdep	MNB-Unigrams
MNB-bigramsPOS	0.450106877	0.199757357
MNB-bigramsPOSVal	0.467056973	0.414459608
MNB-SDep	#DIV/0!	0.713982118
MNB-unigramsPOS	0.309869821	0.091409264
MNB-unigramsPOSVal	0.394189732	0.009052859
MNB-Unigrams	0.2547063	#DIV/0!
MNB-Bigrams	0.765638066	0.909629672
SMO-bigramsPOS	0.317346589	0.052210018
SMO-bigramsPOSVal	0.225232733	0.05661842
SMO-SDep	0.006299444	0.145116503
SMO-unigramsPOS	0.199010638	0.601244279
SMO-unigramsPOSVal	0.328817646	0.388272652
SMO-Unigrams	0.181560125	0.145690068
SMO-Bigrams	0.897983946	0.156656501

Paired t-test (KDD)	Paired t-test (ECML)
MNB-Unigrams	SMO-BigramsPOS
0.023478	0.256616
0.035955	0.19718
0.009691	0.075964
0.106894	0.154811
0.164111	0.137204
#DIV/0!	0.205729
0.007926	0.496865
0.032841	#DIV/0!
0.006623	0.018941
0.000622	0.032399
0.002033	0.055146
0.069784	0.022106
0.142511	0.162038
0.027126	0.28576

Table 44 Full-score regression (left) and ranking (right).

TEST (DOMAIN)	Paired t-test (KDD)	Paired t-test (ECML)
Best model →	SMOrbf-Unigrams	SMOrbf-Unigrams
AR-1000Unigrams	0.11744557	0.47728238
AR-1000Bigrams	0.00688228	0.02968747
Bag-1000Unigrams	0.2490836	0.01385006
Bag-1000Bigrams	0.00106974	0.00430035
SMOrbf-Unigrams	#DIV/0!	#DIV/0!
SMOrbf-Bigrams	0.01802444	0.00053354
SMOreg-Unigrams	0.00333895	0.11499562
SMOreg-Bigrams	0.16838443	0.13340221

TEST (DOMAIN)	Paired t-test (KDD)	Paired t-test (ECML)
Best model →	SMOreg-Unigrams	SMOreg-Unigrams
AR-1000Unigrams	0.800555	0.389159
MNB-Unigrams	0.305198	0.711554
RN-1000Unigrams	0.008219	0.000405
SMO-Unigrams	#DIV/0!	#DIV/0!

B. Significance test regarding out-of domain evaluation.

Table 45 Classification (left) and ranking (right).

Problem	Binary classification	
Unpaired t-test	KDDtoECML	ECMLtoKDD
MNB-bigramsPOS	0.72319964	0.03766439
MNB-bigramsPOSVal	0.11863687	0.00088081
MNB-SDep	0.06410138	0.24014427
MNB-unigramsPOS	0.00028622	0.35581396
MNB-unigrPOSVal	0.00031301	0.28054543
MNB-Unigrams	5.07419E-05	0.95686683
MNB-Bigrams	0.03109718	0.05389798
SMO-bigramsPOS	0.02307777	0.00292870
SMO-bigramsPOSVal	0.03253793	0.0402981
SMO-SDep	0.02642811	0.03608088
SMO-unigrPOS	0.01466253	0.00722113
SMO-unigrPOSVal	0.00892817	0.000234843
SMO-Unigrams	0.00291323	0.08693071
SMO-Bigrams	0.00418236	0.03050103

	Full-score ranking	
Full-score ranking	KDDtoECML	ECMLtoKDD
AR - 1000Unigrams	0.09106	0.147313
MNB - Unigrams	0.000426	0.084518
RN - 1000Unigrams	0.112841	0.447802
SMOreg - Unigrams	0.052346	0.564882

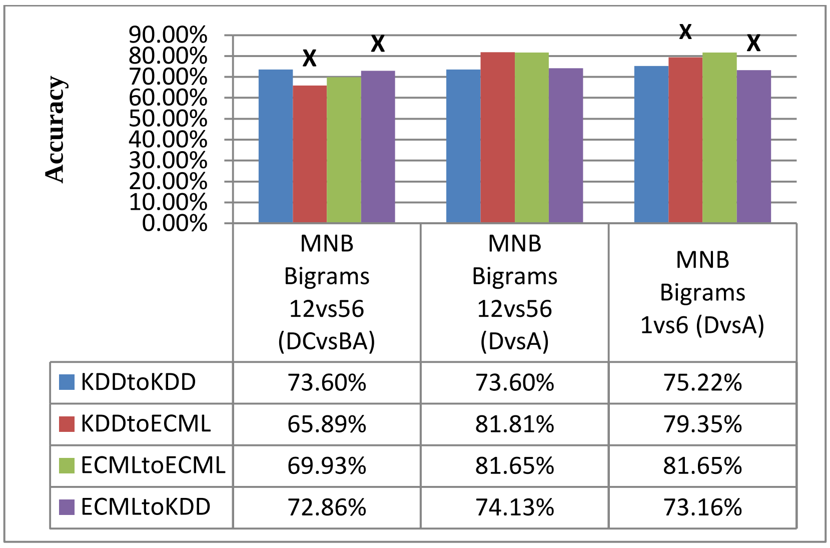
Table 46 Regression.

Problem	Full-score regression		4-score regression		3-score regression	
Unpaired t-test	KDDtoECML	ECMLtoKDD	KDDtoECML	ECMLtoKDD	KDDtoECML	ECMLtoKDD
AR-1000Unigrams	3.68611E-05	7.51629E-09	0.083600846	0.393963	0.498879	0.033843
AR-1000Bigrams	0.000612635	3.36944E-05	0.071194192	0.179609	0.74972	0.107395
Bag-1000Unigrams	2.31015E-08	6.79114E-12	0.006895964	0.057164	0.043803	0.46895
Bag-1000Bigrams	0.000181093	2.81711E-07	0.176127442	0.004513	0.390479	0.010768
SMOrbf-Unigrams	1.76706E-08	2.89851E-08	1.47305E-05	0.0428	0.054999	0.616473
SMOrbf-Bigrams	3.15884E-07	8.60596E-12	0.000410237	0.483738	0.000785	0.192925
SMOreg-Unigrams	2.3767E-08	9.23098E-09	5.97729E-07	0.000942	0.148135	0.681565
SMOreg-Bigrams	4.41099E-09	9.92488E-11	4.05817E-06	0.019126	0.000459	0.0862

APPENDIX G Opinion features and competitive models

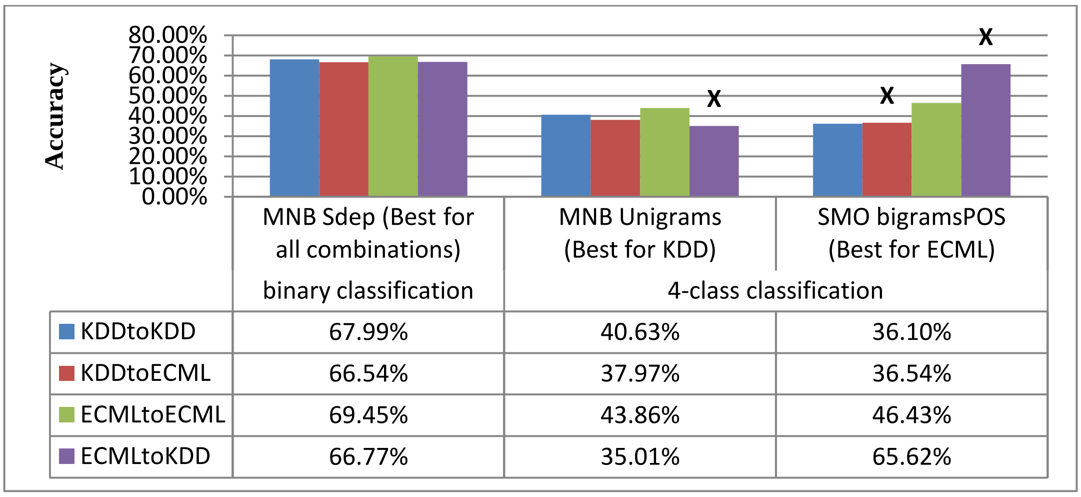
This section summarizes the best models as recommended in the experimental section and also what we can finally learn by these models regarding the phrases that capture opinions in reviews of research papers.

The figures below present the most competitive models per learning problem and conference.



	MNB Bigrams 12vs56 (DCvsBA)	MNB Bigrams 12vs56 (DvsA)	MNB Bigrams 1vs6 (DvsA)
🟦 KDDtoKDD	73.60%	73.60%	75.22%
🟥 KDDtoECML	65.89%	81.81%	79.35%
🟩 ECMLtoECML	69.93%	81.65%	81.65%
🟪 ECMLtoKDD	72.86%	74.13%	73.16%

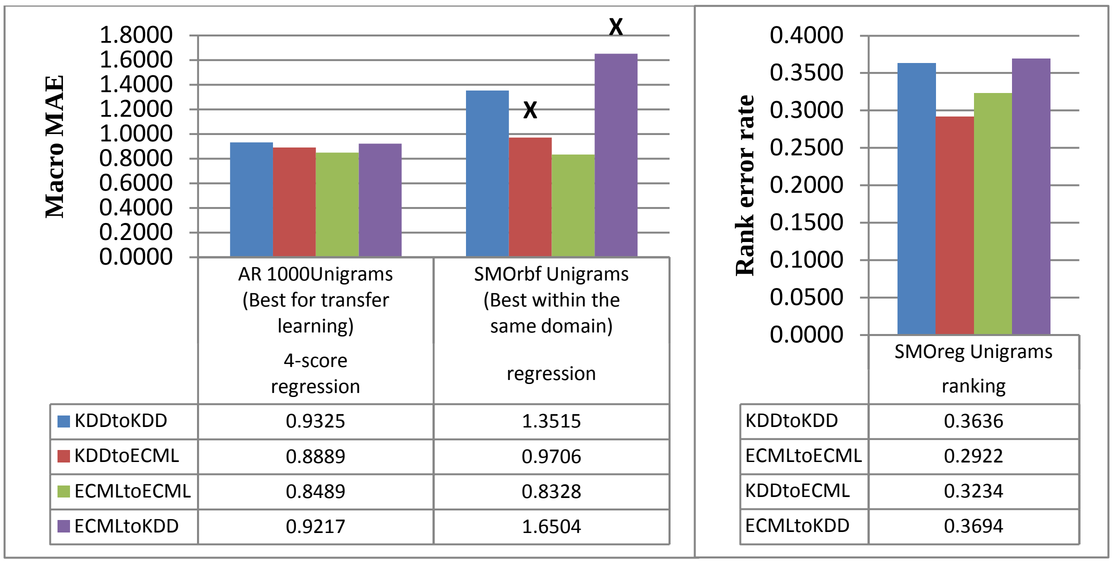
Figure 17 The most similar scores[13] (Section 6.2.2, Table 8).



	MNB Sdep (Best for all combinations)	MNB Unigrams (Best for KDD)	SMO bigramsPOS (Best for ECML)
	binary classification	4-class classification	
🟦 KDDtoKDD	67.99%	40.63%	36.10%
🟥 KDDtoECML	66.54%	37.97%	36.54%
🟩 ECMLtoECML	69.45%	43.86%	46.43%
🟪 ECMLtoKDD	66.77%	35.01%	65.62%

Figure 18 Best models regarding the classification[13].

[13] **X**: Transfer learning is successful (outperforms the baseline) but it is not statistically significant compared to the in-domain performance.



	AR 1000Unigrams (Best for transfer learning) 4-score regression	SMOrbf Unigrams (Best within the same domain) regression
🟦 KDDtoKDD	0.9325	1.3515
🟥 KDDtoECML	0.8889	0.9706
🟩 ECMLtoECML	0.8489	0.8328
🟪 ECMLtoKDD	0.9217	1.6504

	SMOreg Unigrams ranking
KDDtoKDD	0.3636
ECMLtoECML	0.2922
KDDtoECML	0.3234
ECMLtoKDD	0.3694

Figure 19 Best models regarding regression (left) and ranking (right)[13].

The tables below present some examples of opinion features as extracted by different models during the experimental phase.

Table 47 Top 10 unigrams with the highest weight as estimated for regression SMOReg – Unigrams (4-score regression problem).

KDD	**SCORE KDD**	**SCORE ECML**	**ECML**	**SCORE KDD**	**SCORE ECML**
like	0.1053	0.0132	nice	0.098	0.0907
bit	0.1049	0.0379	well	0.0635	0.0904
nice	0.098	0.0907	would	0.0816	0.0796
add	0.0958	0.0174	outperform	0.0097	0.077
scheme	0.085	0.0235	improv	0.0287	0.0676
would	0.0816	0.0796	you	0.0327	0.0666
figur	0.0803	0.0178	good	0.0646	0.066
could	0.0772	0.0261	interest	0.0771	0.0655
interest	0.0771	0.0655	along	0.0181	0.0633
line	0.0766	0.0363	wonder	0.032	0.0623

Table 48 Bottom 10 unigrams with the lowest weight as estimated for regression SMOReg – Unigrams 4 score range.

KDD	**Score**	**ECML**	**Score**
not	-0.1406	not	-0.159
featur	-0.0939	realli	-0.0977
there	-0.0895	onli	-0.0952
what	-0.0883	no	-0.0825
without	-0.082	there	-0.0785
confer	-0.0745	data	-0.0776
specif	-0.0729	what	-0.076
lack	-0.0696	mani	-0.0734
claim	-0.0691	unfortun	-0.0708
compar	-0.067	english	-0.0678

Table 49 Bigrams with the highest and lowest FCE.

HIGH FCE	**SCORE**	**LOW FCE**	**SCORE**
TAGNUMBER TAGNUMBER	4.65423	-rcb- -lcb-	-7.09907
TAGNUMBER and	3.866684	- page	-7.03072
section TAGNUMBER	3.617911	emerging pattern	-6.68571
and TAGNUMBER	2.684131	subgroup discovery	-6.43183
but it	1.960907	they dont	-6.31333
the algorithm	1.925364	$ lpha	-6.24836
the data	1.847804	^ -lrb-	-6.24836
to understand	1.787131	i =	-6.24836
has a	1.677677	p =	-6.24836
is better	1.677677	page TAGNUMBER	-6.24836
as the	1.51479	the encoding	-6.24836
a -rrb-	1.478939	TAGNUMBER paragraph	-6.22339
experimental section	1.478939	couldn t	-6.14223

.

APPENDIX H Code

This section contains a part of the Java code regarding the feature vector representation and the in-domain evaluation process.

```
/**
    * This method creates the data needed to train a classifier and calls
    * the method ProduceModels to train and validate the classifier. This
    * method implements stratified 10 fold cross validation to the dataset.
    *
    *
    */
   public void CreateData(String dataset, String SourcePath, String SourcePathecml, String TargetPath,
String targetPathStatistics, String datasetStringFormat, String datasetStringFormatecml) {

       if (this.modelChoice.equals("SMO")) {
           this.modelChoice = "SVMTuneC";
       }
       try {
           //Prepare arrays for the evaluation measures
           double accuracy[] = new double[10];
           {…}// CODE OMITTED
           double recallPerClass[][] = new double[10][numOfClasses];
           double weightedrecallPerClass[][] = new double[10][numOfClasses];
           double precisionPerClass[][] = new double[10][numOfClasses];
           double overallAccuracy = 0.0;
           double accuracyCV = 0.0;

           //Find the optimal parameters for SVM
           if (modelChoice.equals("SVM_80")) {
               initialiseStatistics();
           }
           Instances Data = null;
           Instances Dataecml = null;

           if (firstRun) {
               if (useFilter) {
                   Data = LoadArff(SourcePath + datasetStringFormat + ".arff");
               } else {
                   Data = LoadArff(SourcePath + dataset + ".arff");}
```

```
                Data.setClassIndex(Data.numAttributes() - 1);
                 Data.randomize(new Random(seed));
                 if (Data.classAttribute().isNominal()) {
                     Data.stratify(10);
                 }
             }

             //Stratified 10-fold cross validation
             for (int j = 0; j < 10; j++)
             {
                 experiment = j;


                 //Feature extraction and feature vector representation if the
                 // the dataset is not in a feature vector form.
                 if (firstRun) {
                     Dataecml = LoadArff(SourcePathecml + datasetStringFormatecml + ".arff");

                     Instances trainInit = null;
                     Instances testInit = null;
                     Instances testInitECML = null;

                     // Merge reviews with specific scores from KDD to ECML
                     if (convertToDifClass) {
                         Data.setClassIndex(0); //wrongly initialise class
                         trainInit = NewFeatures.convertToClass(Data.trainCV(10, j), bigrams);
                         testInit = NewFeatures.convertToClass(Data.testCV(10, j), bigrams);

                        testInitECML = NewFeaturesECML.convertToClass(Dataecml, bigrams);

                    // Merge reviews with specific scores from ECML to KDD
                     } else if (convertToDifClassReverse) {

                         Data.setClassIndex(0); //wrongly initialise class
                         trainInit = NewFeaturesECML.convertToClass(Data.trainCV(10, j), bigrams);
                         testInit = NewFeaturesECML.convertToClass(Data.testCV(10, j), bigrams);
                         testInitECML = NewFeatures.convertToClass(Dataecml, bigrams);
```

```
        // Do not merge anything use the full score range
        } else {
           trainInit = Data.trainCV(10, j);
           testInit = Data.testCV(10, j);
           testInitECML = Dataecml;

        }
        trainInit.setClassIndex(trainInit.numAttributes() - 1);
        testInit.setClassIndex(testInit.numAttributes() - 1);
        testInitECML.setClassIndex(testInitECML.numAttributes() - 1);

        // Feature extraction and vector representation
        if (useFilter) {
           Instances trainInitECML = new Instances(trainInit);
           WekaTrainingSets cDatasets = new WekaTrainingSets();
           Instances[] newData = cDatasets.CreadDataCV(trainInit, testInit);
           Instances filteredTrain = newData[0];
           Instances filteredTest = newData[1];
           cDatasets = new WekaTrainingSets();

           Instances[] newDataECML = cDatasets.CreadDataCV(trainInitECML, testInitECML);
           Instances filteredTestECML = newDataECML[1];
           if (furtherPreprocess) {
              Instances filteredTrainECML = new Instances(filteredTrain);
              WekaTrainingSets cDatasets2 = new WekaTrainingSets();
              Instances[]      furtherFeatures      =      cDatasets2.CreadDataCVfurther(filteredTrain,
filteredTest);
              SaveArff(TargetPath + j + "/" + dataset + "_train.arff", furtherFeatures[0]);
              SaveArff(TargetPath + j + "/" + dataset + "_test.arff", furtherFeatures[1]);
              cDatasets2 = new WekaTrainingSets();
              Instances[]                    furtherFeaturesECML                              =
cDatasets2.CreadDataCVfurther(filteredTrainECML, filteredTestECML);
              SaveArff(TargetPath + j + "/" + dataset + "_ECML.arff", furtherFeaturesECML[1]);
           } else {
              SaveArff(TargetPath + j + "/" + dataset + "_train.arff", filteredTrain);
              SaveArff(TargetPath + j + "/" + dataset + "_test.arff", filteredTest);
              SaveArff(TargetPath + j + "/" + dataset + "_ECML.arff", filteredTestECML);
           }
        } else {

           SaveArff(TargetPath + j + "/" + dataset + "_train.arff", trainInit);
           SaveArff(TargetPath + j + "/" + dataset + "_test.arff", testInit);
           SaveArff(TargetPath + j + "/" + dataset + "_ECML.arff", testInitECML);

        }
```

```
        } // End of feature extraction and feature vector representation

        if (this.FCE && j == 0) {
           dataset = dataset + "FCE" + this.attributesToSelect;
        }

        // Load the training set of the source domain.
        Train = LoadArff(TargetPath + j + "/" + dataset + "_train.arff");
        Train.setClassIndex(Train.numAttributes() - 1);
        // Load the test set of the same domain for in-domain evaluation.
        Test = LoadArff(TargetPath + j + "/" + dataset + "_test.arff");
        Test.setClassIndex(Test.numAttributes() - 1);

        if (this.featureSelection) {
           TestECML = LoadArff(TargetPath + j + "/" + dataset + "_ECML.arff");
           TestECML.setClassIndex(TestECML.numAttributes() - 1);
        }

        //Train a model and apply it to the test set.
        Classifier   cls   =   buildModelINIT(modelChoice,   Train,   Test,   TargetPath   +   j   +
"/Predictions/final/", true);

        if (this.featureSelection) {

           SaveArff(TargetPath + j + "/" + dataset + "IG" + this.attributesToSelect + "_train.arff",
Train);
           SaveArff(TargetPath + j + "/" + dataset + "IG" + this.attributesToSelect + "_test.arff",
Test);
           SaveArff(TargetPath + j + "/" + dataset + "IG" + this.attributesToSelect + "_ECML.arff",
TestECML);
        }

        System.out.println("Experiment :" + j + " finished.");


        // Save the model in order to be able to apply it to a different
        // domain
        if (featureSelection) {
           SerializationHelper.write(TargetPath + j + "/" + modelChoice + "exp" + j + "IG" +
this.attributesToSelect + ".model", cls);
        } else if (FCE) {
           SerializationHelper.write(TargetPath + j + "/" + modelChoice + "exp" + j + "FCE" +
this.attributesToSelect + ".model", cls);
```

```
} else {
    SerializationHelper.write(TargetPath + j + "/" + modelChoice + "exp" + j + ".model", cls);

}
// The model has been saved.

if (featureSelection) {
    Test.deleteAttributeAt(0);
    Train.deleteAttributeAt(0);
    Train.setClassIndex(Train.numAttributes() - 1);
    Test.setClassIndex(Test.numAttributes() - 1);
}

// In-domain evaluation
Evaluation eval = new Evaluation(Train);
eval.evaluateModel(cls, Test);

//Save Confusion Matrix
double[][] cMatrix = eval.confusionMatrix();
try {
    String pathForMatrix = "";
    if (this.featureSelection) {
        pathForMatrix = TargetPath + j + "/" + this.modelChoice + "IG" +
  this.attributesToSelect + experiment + "_cMatrix.ser";
    } else {
        pathForMatrix = TargetPath + j + "/" + this.modelChoice + experiment +
  "_cMatrix.ser";
    }
    FileOutputStream fileOut =
            new FileOutputStream(pathForMatrix);
    ObjectOutputStream out =
            new ObjectOutputStream(fileOut);
    out.writeObject(cMatrix);
    out.close();
    fileOut.close();
} catch (IOException i) {
    i.printStackTrace();
}
//Save Confusion Matrix

//Save the statistical results for this fold.

String summary = eval.toSummaryString("\nResults\n\n", false);
String summary2 = eval.toClassDetailsString();
String summary3 = eval.toMatrixString();
```

```
        totalEvaluation = totalEvaluation + summary + summary2 + summary3;

        auc[j] = eval.weightedAreaUnderROC();
        precision[j] = eval.weightedPrecision();
        recall[j] = eval.weightedRecall();
        tpRate[j] = eval.weightedTruePositiveRate();
        {….} //CODE OMITTED
        int numClasses = Test.numClasses();

        for (int classIndex = 0; classIndex < numClasses; classIndex++) {
            weightedrecallPerClass[j][classIndex] = eval.weightedRecall(classIndex);
        }
        for (int classIndex = 0; classIndex < numClasses; classIndex++) {
            recallPerClass[j][classIndex] = eval.recall(classIndex);
            precisionPerClass[j][classIndex] = eval.precision(classIndex);
        }

        accuracy[j] = (eval.correct() / eval.numInstances()) * 100;
        overallAccuracy = overallAccuracy + accuracy[j];

        // Statistical results for this fold of 10-fold CV have been updated.

    }

    //Save the statistics from every fold to a .txt file
    accuracyCV = overallAccuracy / 10.0;
    System.out.println("FinalAccuracy = " + accuracyCV);
    saveStatistics(accuracy, targetPathStatistics, dataset);
    {….}

    }
    // Statistics have been saved

  } catch (Exception ex) {
    System.out.println(ex);
    ex.printStackTrace();
  }
}
```