2014-5

Dynamic Estimation of Rater Reliability using Multi-Armed Bandits

Alexey Tarasov
Technological University Dublin, tarasovsaleksejs@gmail.com

Dynamic Estimation of Rater Reliability using Multi-Armed Bandits

Alexey Tarasov

A thesis submitted to the Dublin Institute of Technology

in fulfillment of the requirements for the degree of

Doctor of Philosophy

School of Computing

May 2014

Declaration

I certify that this thesis which I now submit for examination for the award of Doctor of Philosophy, is entirely my own work and has not been taken from the work of others save, and to the extent that such work has been cited and acknowledged within the text of my work.

This thesis was prepared according to the regulations for postgraduate study by research of the Dublin Institute of Technology and has not been submitted in whole or in part for an award in any other third level institution.

The work reported on in this thesis conforms to the principles and requirements of the DIT's guidelines for ethics in research.

DIT has permission to keep, lend or copy this thesis in whole or in part, on condition that any such use of the material of the thesis by duly acknowledged.

Signature _____ Date _____

Acknowledgements

I would like to thank Science Foundation Ireland (Grant No 09-RFP-CMS253) and Dublin Institute of Technology (Fiosraigh scholarship) for their financial support over the last four years, and the opportunity to devote myself solely to my research.

This thesis would not have been completed without the guidance and support of my supervisors, Dr Sarah Jane Delany and Dr Brian Mac Namee. I wish to thank them for all the feedback, advice and knowledge they gave me. I am also grateful to Dr Charlie Cullen and Prof. Pádraig Cunningham, who supervised me during the early stages of my studies at DIT.

Special thanks go to all my colleagues from the Digital Media Centre and Applied Intelligence Research Centre, especially Dr Viacheslav Filonenko, John Snel, Dr Brian Vaughan, John McGee, Niels Schütte and Dr Mark Dunne. One cannot ask for better colleagues than these guys. I also owe my thanks to Dr Rong Hu for allowing me to use some of her code, and Anna Deegan, who helped us a lot with designing the interface for rating emotions. Very special thanks go to the people from the Graduate Research School, especially Gerolmina Di Nardo for her help and support. I would also like to thank Dr Richard Southern and other colleagues of mine from Deloitte Analytics for their great support.

A specific feature of machine learning research is that it depends heavily on datasets, which are usually tricky to get. I wish to express my thanks to Dr Ben Leong, Dr Martha Larson, Dr Rada Mihalcea, Dr Malcolm Slaney and Dr Mohammad Soleymani for kindly allowing me to use their data. The datasets they generously shared with me allowed me to make the experiments of this thesis far sturdier. Also, without the effort of many raters,

anonymous or not, checking many hypotheses from this thesis would have been impossible.

During my studies, many people allowed me to help them with their research. Although such "side quests" were not always directly linked to the topic of my thesis (which, of course, annoyed my supervisors a bit), they allowed me to learn a lot about research in general, and some specific techniques in particular. I wish to thank Dr Alexei Pozdnoukhov, Felix Kling, Dr Ibrahim Saygin Topkaya, Mehmet Umut Şen, Mustafa Berkay Yilmaz, Prof. Hakan Erdoğan, Dr Marcin Włodarczak and Catharine Oertel.

I would never even have thought about doing a Ph.D. if I had not met many brilliant academics during my undergraduate studies at the Transport and Telecommunication Institute in Riga. I am deeply indebted to the supervisors of my B.Sc. and M.Sc. theses, Prof. Eugene Kopytov and Prof. Vladimir Labendik, for awakening my interest in applied machine learning. I also wish to express my thanks to Prof. Irina Yatskiv and Prof. Arkady Borisov for being great mentors not only when I was a student, but also when I was making my first steps as an assistant lecturer.

My friends and family are among the people who deserve my warmest gratitude. I would especially like to mention Alexey Chuenko, Leonid and Natalija Alexandrovs, Simon and Julia Yakunins, Nikolaj and Anastasija Karpovs as great supporters during different stages of my life, including the difficult transition of moving to a different country. I also thank my parents, Svetlana and Vladimir, for their permanent support, encouragement and for always being there for me. And lastly, I wish to thank Stephen for being the best proofreader and partner that any person could ever want.

Alexey Tarasov

Dublin Institute of Technology

May 2014

Abstract

One of the critical success factors for supervised machine learning is the quality of target values, or predictions, associated with training instances. Predictions can be discrete labels (such as a binary variable specifying whether a blog post is positive or negative) or continuous ratings (for instance, how boring a video is on a 10-point scale). In some areas, predictions are readily available, while in others, the effort of human workers has to be involved. For instance, in the task of emotion recognition from speech, a large corpus of speech recordings is usually available, and humans denote which emotions are present in which recordings. Crowdsourcing is a commonly used technique in such tasks. However, one of the main (and still unresolved) challenges in crowdsourcing is the presence of unreliable workers–people who work without investing real effort, or those who do not possess sufficient skills for the task at hand. If the task is labelling/rating training data, the presence of unreliable workers may result in low quality target values. A typical quality control measure is to collect a large number of ratings for each instance. While this is feasible in cases where the price for a single rating is low, this approach becomes less practical with more complicated tasks that entail a high price for a single rating. In such conditions, it makes sense to track rater reliability dynamically, as they rate.

Although many such approaches exist, they often make unrealistic assumptions that make their use in real life complicated if not impossible. The aim of this thesis is to develop an approach to dynamic estimation of rater reliability, based on multi-armed bandits, that can be used in real-life crowdsourcing tasks. The field of emotion recognition from spoken speech is used as a motivating example throughout the thesis, although all approaches are also tested on data from other domains. The algorithms proposed in the thesis are

primarily targeted at tasks where predictions are continuous or ordinal, but they can also be easily generalised for binary or multi-class classification.

This thesis first examines the problem of dynamic estimation of reliability in simplified conditions of constant rater availability, namely, assuming that any rater is permanently engaged in the rating process and can provide a rating instantaneously. The evaluation shows that two multi-armed bandit approaches—ϵ-first and KL-UCB—can outperform a state-of-the-art baseline IEThresh, as well as a naïve algorithm that gathers ratings from random raters. Additionally, we also explore the bootstrap problem, i.e. the problem of the low accuracy of ratings gathered at the initial stage of the rating process, when rater reliability cannot yet be estimated precisely. Our experiments suggest that gathering additional ratings for training instances rated at this initial stage can improve the accuracy of predictions.

The second approach proposed in this thesis, DER^3 (Dynamic Estimation of Rater Reliability for Regression), works in more realistic conditions of intermittent rater reliability, i.e. when every rater can leave and re-enter the rating process at arbitrary times. The main idea of the approach is that when a rater becomes available, an instance is selected where a rating by the current rater would be of the most benefit, and the rater is asked to rate it. The results of our experiments strongly suggest that DER^3, based on ϵ-first and KL-UCB multi-armed bandits, can improve the quality of predictions compared to a baseline where ratings are accepted from all incoming raters.

The last major contribution of the thesis is the validation of DER^3 on the Amazon Mechanical Turk platform for the task of rating a corpus of emotional speech. DER^3 facilitates getting predictions of a higher quality without incurring any additional cost.

Contents

List of Figures

List of Tables

Associated Publications

The publications that are related to this thesis are listed below:

1. Tarasov, A. and Delany, S.: 2011, Benchmarking Classification Models for Emotion Recognition in Natural Speech: A Multi-corporal Study, *Proceedings of IEEE International Conference on Automatic Face & Gesture Recognition and Workshops*, pp. 841–846.

2. Tarasov, A., Cullen, C. and Delany, S.: 2010, Using Crowdsourcing for Labelling Emotional Speech Assets, *Proceedings of W3C Workshop on Emotion Markup Language.*

3. Tarasov, A., Delany, S. and Mac Namee, B.: 2012, Dynamic Estimation of Rater Reliability in Regression Tasks using Multi-Armed Bandit Techniques, *Proceedings of Workshop on Machine Learning in Human Computation and Crowdsourcing, in conjunction with ICML.*

4. Snel, J., Tarasov, A., Cullen, C. and Delany, S.: 2012, A Crowdsourcing Approach to Labelling a Mood Induced Speech Corpus, *Proceedings of the 4th International Workshop on Corpora for Research on Emotion Sentiment & Social Signals, in conjunction with LREC.*

5. Tarasov, A., Delany, S. and Mac Namee, B.: 2012, Dynamic Estimation of Rater Reliability in Subjective Tasks Using Multi-Armed Bandits, *Proceedings of ASE/IEEE International Conference on Social Computing*, pp. 979–980.

6. Tarasov, A., Delany, S. and Mac Namee, B.: 2013, Improving Performance by Re-Rating in the Dynamic Estimation of Rater Reliability, *Proceedings of Machine*

Learning Meets Crowdsourcing Workshop, in conjunction with ICML.

7. Tarasov, A., Delany, S. and Mac Namee, B.: 2014, Dynamic Estimation of Worker Reliability in Crowdsourcing for Regression Tasks: Making It Work, *Expert Systems with Applications* **41**, pp. 6190–6210.

Chapter 1

Introduction

In the last few years the "wisdom of crowds" has become more and more evident. The Internet facilitates access to a great variety of people with different skills and backgrounds, and this can be used to get fast and cost-effective input into solving different problems. Using a large number of people over the Internet to solve some tasks is called *crowdsourcing*. Crowds help to solve complicated scientific problems on web sites such as Kaggle[1] or InnoCentive[2], but online workers would also happily engage in something more mundane such as the translation of phrases from one language to another or image tagging. Such tasks are usually paid for, so doing them becomes a profitable hobby or even a full-time job for many people. There are numerous crowdsourcing initiatives that have made a difference in the world. For instance, the MySmartEye[3] smartphone application allows a blind person to take a picture of some object, such as a jar of sauce, and send it to the crowd to serve as "eyes", responding with the cooking instructions. Initially, there might have been a certain scepticism associated with large, chaotic crowds, but now crowdsourcing is becoming an accepted technique to use in business. A recently released Deloitte report "Tech Trends 2014"[4] lists industrialised crowdsourcing among ten topics that "are transforming business, government and society".

One of the areas that is undergoing such a transformation is supervised machine learn-

[1] https://www.kaggle.com/
[2] https://www.innocentive.com/
[3] http://mysmarteye.starhub.com/
[4] http://www2.deloitte.com/ie/en/pages/technology/articles/Tech-Trends-2014.html

ing, which is basically learning based on examples. For instance, if the task at hand is to train an automatic classifier which determines whether a certain e-mail is spam or not, a corpus of e-mails marked with discrete *labels* such as "Spam" or "Not spam" is required. Quite often, such labels come not from a predefined set of classes, but are numeric values. For instance, researchers often encounter rating tasks, where a label for a training instance is number on a certain scale. This number corresponds to the degree of certainty with which a given property manifests itself in that training instance, for instance, how boring a video is, or how funny a joke is, and so on. In this thesis, we refer to such labels as to *ratings*. A corpus of training instances is often regarded to as a *training set*.

Researchers' attention has long focused on how to represent *training instances* (e.g. e-mails in the spam recognition scenario) as numeric vectors, and which classification technique to use. However, there are many domains where collection of labels for the training data, such as "Spam" or "Not spam", is a complicated task on its own. For example, in emotion recognition from speech the task of supervised machine learning is to recognise an emotion that is expressed by a speaker in a short speech clip. When it comes to labelling/rating such training data, it is not even widely agreed whether emotions should be perceived as distinct categories (such as anger or joy) or whether it is better to use continuous rating scales such as "how active the person is on the scale from 0 to 1". But even if there are no questions about the annotation scheme, there is still the problem of subjectivity: different people can label/rate the same clip differently.

One way of coping with this is to use output from several *raters* or *labellers*. When several ratings for each training instance are gathered, they are aggregated in some way (for instance, averaged) to obtain a single value for each instance. This value, or *prediction*, can then be used to train a classifier or predictor. In this way, working with the output from multiple labellers (which is often performed via crowdsourcing) often becomes a distinct step in the supervised machine learning process. This thesis is devoted to this step, namely the collection of ratings for data to be used in supervised machine learning using crowdsourcing.

One common problem with crowdsourcing in general is the presence of raters who

provide noisy ratings due to insufficient skill or the intention of getting paid without actually investing any effort. In a supervised machine learning context it means that the resulting ratings/labels will be inaccurate and can negatively impact the quality of the classifier. Averaging ratings (or taking a majority vote, in case of labels) reduces this effect, however, many researchers show that the accuracy of predictions can be improved by taking rater reliability into account [136, 198]. There are a few ways to do this. For instance, if any data about the previous performance of raters is available, raters who did not provide accurate results in the past are simply not allowed to participate. However, this option is infeasible if such information is not available. Under these circumstances quality control can only be based on the actual ratings provided. A quality control technique can be *static* or *dynamic*.

A static quality control procedure first gathers N ratings for every training instance from any available raters, and then estimates rater reliability via an expectation-maximisation algorithm [136]. At the end of the process, both rater reliabilities and predictions are calculated iteratively. The end result is typically numeric reliabilities of raters: the higher the number, the more reliable the rater. Predictions are then calculated as an average rating weighted by reliability, so that ratings from unreliable raters receive low weights and have a minimal impact on predictions. In contrast, dynamic quality control techniques estimate rater reliability as raters rate, asking only those raters who were reliable to date [55]. Usually N ratings are gathered for each instance. In the static scenario, all instances are usually put online, and when a rater becomes available, an instance is chosen randomly, and the rater is asked to provide a rating. Each rater can rate as many instances as desired. In the dynamic scenario however, such an approach can cause problems. When rater reliability is estimated dynamically, the quality of every rating is evaluated. As the correct ratings, or *gold standard*, are not available, the rating coming from a particular rater is typically compared to the ratings submitted by other raters. This means that it is important to get at least a small number of training instances rated by multiple raters. Achieving such coverage quickly enough with random assignment of ratings is not possible. Hence, some other means of assigning ratings to instances is required. For example,

an instance is selected and all raters who become available are asked to rate it, until the required number of ratings is collected [197].

The main problem with static techniques is that they result in paying for ratings that, in the end, are not used as they are assigned with low weights. Dynamic techniques, however, provide better "value for money", as noisy ratings are rarely accepted. Ideally, a dynamic technique gathers only ratings that can actually be useful, and avoids paying noisy raters. This reasoning is especially important for tasks where the price for a single rating is high. In such contexts, it is especially critical to pay for as few ratings as possible, while achieving high prediction quality. Dynamic techniques for the estimation of rater reliability are the main focus of this thesis.

Although a number of dynamic techniques exist, they often suffer from limitations that make their usage in practice complicated if not impossible. For instance, such techniques often are suited only to very specific types of task, such as binary labels [79, 201, 216] and can not be easily adapted to multi-class labels or ratings. Some algorithms [34, 181] assume that the quality of every rating can be estimated instantly and independently of other ratings. In supervised machine learning, this is equivalent to assuming that there is an oracle that can always provide a correct rating. However, if such an oracle is available, there would be no need to collect ratings in the first place [79]. Many dynamic techniques [93, 197] also require quite specific knowledge about the task prior to the rating process, e.g. the statistical distribution of rater errors [93, 197], while this knowledge is rarely available before the rating process starts. Additionally, the majority of dynamic techniques assume that raters are permanently engaged in the rating process, i.e. they can provide a rating immediately upon being asked [33, 47, 55, 129, 181, 195, 201, 205]. Such conditions, which we refer to as *constant rater availability* can arise for some tasks. However, in many scenarios (for instance, on a platform similar to Amazon Mechanical Turk) workers enter and leave the rating process at undefined times. Such *intermittent availability* is more realistic, but there are relatively few dynamic approaches that can work in such conditions [34, 197, 216].

The main contribution of this thesis is the Dynamic Estimation of Rater Reliabil-

ity for Regression (DER^3) approach that is specifically suited for real-life crowdsourcing scenarios, and is free from the limitations of the state-of-the-art dynamic techniques highlighted above. Also, as with any dynamic technique, it facilitates reaching low prediction error, while keeping costs down. The approach has been evaluated using both simulated experiments and a real-time evaluation using the Amazon Mechanical Turk platform.

The DER^3 approach represents a rating problem as a *multi-armed bandit*. A multi-armed bandit is a mathematical abstraction that represents the task of choosing between alternatives as a multi-armed gambling machine. Each arm represents a single alternative, and the goal is to find the best arm(s) in as few pulls as possible. In DER^3 each arm corresponds to a rater, while pulling an arm is equivalent to asking a rater to provide a rating.

Additionally, a few associated issues were investigated in the thesis, including a bootstrapping problem associated with using any exploration-exploitation algorithm such as a multi-armed bandit. The problem manifests itself in the very beginning of the rating process, when rater reliabilities have not been estimated precisely enough. This means that noisy raters are asked to rate quite often, which has a negative impact on predictions. The approach suggested in this thesis is to gather additional ratings for such instances from raters who are known to be reliable, at the end of the process. These and some other contributions are briefly summarised in the next section.

1.1 Contributions of the thesis

In this work, a particular emphasis is placed on dynamic quality control in crowdsourcing using multi-armed bandits. The field of emotion recognition from speech is used as the motivating example, as rating training data in this area requires output from multiple raters. The aim of the thesis is to develop an approach to estimate the rater reliability dynamically for rating corpora to be used in supervised machine learning for the conditions of intermittent rater availability.

The key contribution in this thesis is the DER^3 (Chapter 7) that can be applied for

real-life crowdsourcing tasks, unlike the majority of state-of-the-art algorithms, which often have practical limitations. The main features of DER^3 are the following:

1. It is suited for a broad variety of tasks, including regression, multi-class and binary classification.

2. It works in the conditions of intermittent rater availability, i.e. when raters can enter and leave the process at arbitrary times.

3. It does not require any prior knowledge about the task, such as distribution of rater errors.

4. It works in the conditions when the quality of a single rating can not be independently verified.

5. It does not demand that training instances have features[5] associated with them.

Another important contribution is the validation of the DER^3 approach using Amazon Mechanical Turk (Chapter 8).

Additionally, there are a few supporting contributions in this thesis. They are the following:

- **An approach to dynamic estimation of rater reliability in the scenario of constant rater availability (Chapter 5)** served as a feasibility study for using multi-armed bandits in dynamic estimation of rater reliability. It operates in simplified conditions, where every rater is immediately available to provide a rating at any time.

- **Investigation into handling the bootstrap issue in the scenario of constant rater availability (Chapter 6):** an approach to improving the quality of predictions by acquiring additional ratings for training instances rated at the stage of exploration when bootstrapping takes place.

[5]Numerical or discrete values, representing an instance as a set of values that are used in supervised machine learning.

Table 1.1: Contributions, corresponding chapters and publications.

Contribution	Chapter	Publications
DER^3, a novel approach to dynamic estimation of rater reliability in the scenario of intermittent rater reliability	Chapter 7	[178]
Real-life evaluation of dynamic estimation of rater reliability based on multi-armed bandits	Chapter 8	[164, 173]
Approach to dynamic estimation of rater reliability in the scenario of constant rater reliability	Chapter 5	[175, 176, 178]
Investigation into handling the bootstrap issue in the scenario of constant rater reliability	Chapter 6	[177]
Contextualisation of dynamic approaches	Chapter 3	–
Benchmark of supervised classification techniques on emotional speech data	Appendix A	[174]

- **Contextualisation of dynamic approaches (Chapter 3):** different ways of classifying crowdsourcing tasks were considered, and an overview prepared. As a result, the characteristics of rating problems were identified. Then the recommendations about choosing a static or dynamic quality control technique were formed on the basis of these characteristics, as well as some newly proposed categories.

- **Benchmark of supervised classification techniques on emotional speech data (Appendix A):** as with any dynamic technique, the DER^3 approach needs to determine a sequence in which training instances are presented to raters. Following the state-of-the-art research [55], active learning was used in this thesis. However, a classifier of predictor is required to use active learning. In the emotion recognition community there is no consensus on which supervised machine learning technique is the most accurate for this domain. That is why a separate experiment to determining such technique was conducted. Multiple natural speech datasets were used to benchmark the performance of a selection of state-of-the-art classification techniques that are used in emotion recognition from speech.

1.2 Summary and structure of this thesis

The remainder of this thesis is structured as follows. Chapter 2 briefly covers the main concepts of supervised machine learning using emotion recognition from speech as a motivating example, as well as describes how labelling of the training data happens in this application area. Chapter 3 is devoted to crowdsourcing, which is a natural way of collecting ratings in research into emotional speech. This chapter contains a detailed overview of quality control measures and taxonomies for crowdsourcing tasks. The conclusion is that none of the existing taxonomies can successfully be used for rating tasks, hence, some new categories were proposed and illustrated by a few examples. Multi-armed bandits, the main technique that is used in this thesis to estimate rater reliability dynamically, is also covered. Chapter 4 describes the methodology for the experiments conducted in this thesis. The approach to estimate rater reliability dynamically for constant rater availability is proposed and evaluated in Chapter 5. Chapter 6 looks into handling the bootstrap issue in the case of constant rater availability. Chapter 7 covers the conditions of intermittent rater availability: in it the DER^3 approach is proposed and evaluated. Chapter 8 evaluates the DER^3 on Amazon Mechanical Turk platform. Chapter 9 concludes the thesis, summarises the conclusions and proposes directions for future research.

Chapter 2

Rating Training Data for Supervised Machine Learning

Supervised machine learning has several distinctive phases, including collection of instances to be used in training, extraction of features, training the classifier/predictor and evaluating its performance. This thesis focuses only on one phase, namely, rating data to be used in training. In this chapter we use our motivating example—emotion recognition from speech—to illustrate all the concepts mentioned. In this domain most researchers use multiple people to rate speech clips in such a way that a resulting rating for a recording is a combination of ratings submitted by multiple raters. This makes the rating of an emotional speech task that by definition requires using multiple raters.

This chapter is structured as follows. Section 2.1 describes the process of supervised machine learning, briefly mentioning all its stages. Section 2.2 describes the details of rating emotions, while Section 2.3 concludes the chapter.

2.1 Overview of supervised machine learning

Alpaydin [3] proposes the following mathematical formulation of supervised machine learning problem. Let us consider that there are K *training instances* that comprise a training set $X = \{\mathbf{x}^t, p^t\}_{t=1}^{K}$. Every training instance is a vector of M numerical values that characterise this instance and represent certain properties of it. These values $\mathbf{x} = [x_1, x_2, ..., x_M]$

are called *features* . The goal is to find a function f such that $f(\mathbf{x}^t) = p^t$ for $t = 1, 2, ..., K$. In real life it often is impossible, as there can be some information about training instances that is not represented in features, so $f(\mathbf{x}^t) + \epsilon = p^t$, where ϵ is random noise. If p^t is a continuous variable, the task at hand is regarded as *regression*. If p^t values are from a discrete set, the task is called a *classification* task. Usually p^t is called *a prediction*.

Although supervised machine learning is used in many different domains, it typically consists of the following steps:

1. **Data acquisition** that consists of getting data that will be used for training. In the context of emotion recognition, it is a corpus of emotional speech recordings, and the overall process depends very much on its quality. There are numerous challenges with getting real, non-acted emotions, as well as making sure that all necessary emotions are presented. There are three kinds of emotional speech corpora: acted (actors are hired to depict certain emotions), natural (recordings from some existing source such as a talk show or call centre) and elicited or induced (people are placed in a controlled environment that provokes them to exhibit certain emotions naturally). Comparing the complexity of the task of automatic emotion recognition from speech, the state-of-the-art research unambiguously states that it increases with the naturalness of instances [16, 102, 192], i.e., it is much simpler for machine learning techniques to recognise acted speech than natural.

2. **Feature extraction and selection**: once the training data is collected, there is a need to extract vectors $\mathbf{x}$ from them. In some areas this process is very straightforward, but it is not so for emotion recognition. There are hundreds and thousands of features that can potentially be derived from an audio signal, some of them are acoustical or spectral, while others can represent lexical information, i.e. what has been said in a phrase.

3. **Data labelling**: when training instances are acquired, there is a need to rate them with proper p^t values that are called *labels*. If labels are numerical values on certain scale, the process is often called *data rating*, while the values themselves are regarded

as *ratings*. In emotion recognition from speech, labels/ratings represent emotion expressed in speech recordings. Currently there is no single widely accepted schema for rating emotions, but two main approaches exist. One proposes that there is a finite set of discrete emotions and that categorical labels can be used to label them [52, 51, 56, 143]; another insists on the existence of infinitely many emotions and uses a dimensional system to rate them [70, 71, 145, 184]. The way people perceive emotions is very personal, and presents a further challenge. Generally, the efforts of many raters are required. Labels or ratings given by many raters can often contradict each other, so there is a need for techniques that would measure the amount of disagreement, and combine all these labels into a single one that will be associated with particular instances.

4. **Training a classifier/predictor**: this stage requires a training set to proceed with training a model that will perform the actual recognition, i.e. find the function $f(\mathbf{x})$. Many techniques that are used in other areas are also exploited for emotion recognition from speech, but it is still unknown which of these is the most accurate. Here are some of the algorithms that are most widely used in emotion recognition from speech:

   (a) Artificial neural networks including two-layered perceptrons [16, 17, 20, 30, 81, 100, 120, 171, 194] and radial basis function artificial neural networks [81].

   (b) k nearest neighbours [7, 6, 53, 62, 73, 72, 89, 120, 126, 125, 149, 160, 208, 209, 194]

   (c) Naïve Bayes [7, 20, 6, 5, 53, 62, 81, 89, 88, 192]

   (d) Decision trees: C4.5 [6, 5, 52, 53, 81, 89, 88, 120, 149], C5.0 [98], ID3 [6, 5] and NBTree [7, 5].

   (e) IF-THEN rules in the form of decision tables [53, 89], PART [53, 81, 89], Ridor [81] and nearest neighbor generalization [53, 81].

   (f) Support Vector Machines (SVM) [20, 102, 53, 62, 88, 98, 97, 149, 146, 148, 154, 157, 160, 169, 179, 189, 190, 192, 207, 209] with mostly radial basis function

[52, 120, 169, 214] and linear kernels [147, 106, 210]. In the task of emotion recognition usually there are more than two classes, that is why a lot of researchers use ensemble techniques—one-to-many ensemble [4, 44], round-robin ensemble [4, 124] or similar approach when SVMs are assembled in a form of a tree [150]. Sometimes the number of SVMs in the round-robin ensemble can be reduced, taking into consideration only those emotions that are close [206, 210]. Some experiments have shown that these two approaches do not exhibit any significant difference in the practice of emotion recognition [155]. There also exists the third alternative—so-called SVM-tree [150] which classifies an input utterance by "one vs. one" classifiers layer-wise, until only one class remains. [89, 120, 150]. Support Vector Regression (SVR) is also used in the field of emotion recognition when numerical scales are used to rate [71, 73].

(g) Gaussian mixture models [108, 122, 124, 138, 155, 162, 170, 183, 193].

(h) Hidden Markov models [10, 28, 99, 125, 151, 180, 151, 159, 191].

(i) Ensembles including boosting [52, 53, 81, 89, 110, 111, 149, 160, 184, 207, 212], bagging [53, 81, 149], random forests [81, 120, 152], stacking [149] and additive trees [52]. One of the most popular techniques used as a base classifier are C4.5 decision trees [89, 149, 160, 207]. Used as a base classifier, C4.5 performs very closely to PART and decision table [89]. Some researchers use a combination of different classifiers like SVM, Naïve Bayes, C4.5 and k nearest neighbours (k-NN) [149] or create an ensemble of hidden Markov models (HMMs) [212].

The task of comparing a variety of techniques on multiple datasets has not been widely explored. That said, some research reports that SVMs outperform other methods by as much as 17% [208], or that they are the most accurate classifier [62]. Other evidence proves that the performance of most of the techniques mentioned in this section is very similar. Ensembles, decision trees, artificial neural networks, SVMs and k nearest neighbours perform best, and have nearly the same performance figures in nearly all research that compares different techniques [20, 89, 120, 160, 207].

However, there is some work where k nearest neighbours and decision trees [149] had the poorest accuracy.

Naïve Bayes is a consistently poor performer, sometimes having a recognition rate 10% worse than the next worst classifier [149], and it is sometimes very close to other techniques that performed very badly [20, 62, 5, 6]. We are aware of just one corpus where Naïve Bayes performance was close to k nearest neighbours and decision trees [7, 6, 5].

It is interesting to note that variations in the performance of one algorithm using different features is often bigger than between different algorithms. For instance, artificial neural networks can perform 25–35% worse than artificial neural networks using different features [30, 100, 128, 194], but their performance will be only 2% worse then the best classifier in cases where identical features are used [120].

Section 2.2 is devoted to data labelling, which is the focus of this thesis.

2.2 Data labelling

In supervised machine learning problems, getting labels for training data is addressed in different ways. In some tasks, the ratings (or labels) are available immediately, for instance, if the task at hand is prediction of whether a certain customer of an insurance company will lapse, i.e. stop having insurance policy by, for instance, moving to a different company. Training data might consist of detailed histories of customers, and it is retrospectively known whether a certain policy lapsed. In other domains, such as medical imaging, ratings are not so readily available. For instance, if the goal is to train a computer-aided diagnostic system that determines whether a lesion is malignant or benign, several qualified radiologists would have had to go through all the images to be used in training, and rate them accordingly [136]. Such redundancy is required to compensate for occasional mistakes that a single radiologist can make due to lack of experience with a particular type of lesion.

Getting ratings can involve complicated collection processes; however, it is not the

only challenge. In many areas, including emotion recognition from speech, there is no single widely-accepted way to label or rate instances. For instance, there is no consensus on a number of emotions, or even the existence of discrete emotions at all. The rest of the section describes how such challenges are addressed by state-of-the art researchers in emotion recognition from speech. The main approaches to selecting raters are described in subsection 2.2.1, while subsections 2.2.2 and 2.2.3 cover two main ways to label or rate emotions.

2.2.1 Selection of raters

One of the most straightforward approaches to rating instances is the self-report, where the person presented in the recording reports emotions felt. Although it is used in some state-of-the-art research [51, 184, 149], the conclusions regarding its validity found in literature contradict one another. Some research claims that this approach is very unreliable, because it is not clear if subjective feeling correlates with other indicators of emotion, including voice indicators [39]. Other researchers declare that it is valid when it is applied right after the recording was made [119]. The main problem with self-reporting is that there are individual differences in awareness of and willingness to report on emotional states that potentially compromise the individual. For instance, people may not report extreme anger or similar emotions that are considered rude or impolite. Nevertheless, self-reporting could be used for selecting labels to use in the labelling process [53], if not for actual labelling.

Considering the issues with self-reporting, the need for specially designated raters is evident. It is not widely agreed whether raters should be experts in emotion recognition, and we are not aware of any research that compares such experts and naïve labellers. Some research has used experts [2, 122, 189], while some has used non-experts [162, 30], but most often the researchers do not make any remarks in this regard.

It is also not widely decided what is the optimal number of labellers. The only suggestion for it as far as we know is ten, but as ten requires too much effort, three is the minimum (mostly for the reason of ensuring the possibility of majority decision) and five is a good compromise [18]. In the state-of-the-art research this number is at least two

[2, 51] and very often lies in the range from three to nine [30, 98, 107, 122, 162]. As a very rare exception bigger numbers of raters also can be encountered—for example, twenty-six [89, 88] and seventy-two [115].

2.2.2 Discrete labels

This subsection describes how discrete labels are used in the task of emotion recognition from speech. We elaborate on how labels are chosen, as well as on how the output of many labellers could be combined to produce a single label for the instance. A few ways of measuring the degree of agreement between labellers, also are presented.

Choosing the labels

The approach of labelling emotions with discrete labels presumes that there exist some number of discrete emotions that can be separated from each other easily. It is mostly based on a theory of basic emotions that are defined as being inborn and universal reactions [45]. Version of Ekman et al. [56] of this set is joy, sadness, fear, anger, surprise and disgust; anticipation and acceptance are added to this set in a model called Plutchik wheel [143]. These are often regarded as primary emotions, and secondary emotions are combinations of these—for example, love is a combination of joy and acceptance and submission is a combination of acceptance and fear. Additional emotions can be classified as presenting different degrees of intensity of primary and secondary emotions, for example, anger can range from annoyance to rage [52].

While performing labelling of instances, two strategies are possible—providing a set of labels and asking raters to use it or perform free annotation, where labellers can choose any labels they want. The latter usually leads to a very large amount of labels (176 in some research [52]) and produces much lower agreement [39] between raters, but can be used prior to labelling to select labels [52]. That is why most of the state-of-the-art work in emotion recognition follows the first strategy. Table 2.1 summarizes information about the labels most often used in some emotional corpora, each row corresponding to one corpus that often is utilized in many publications. Table 2.2 contains information about

Table 2.1: Labels that are most often used in state-of-the-art research in emotional speech recognition. Each row corresponds to a single corpus

References	Anger	Boredom	Disgust	Fear	Happiness	Joy	Negative	Non-negative	Neutral	Positive	Sadness	Surprise
[7, 6, 5]	+		+	+		+			+		+	
[10]	+			+	+				+		+	
[27]	+	+	+	+		+			+		+	
[35]	+		+	+	+						+	+
[44, 121, 160, 146, 148, 152]	+				+				+		+	+
[44, 144, 146, 148, 152, 155, 160, 194, 191, 204]	+	+	+	+	+				+		+	
[102]	+				+				+		+	+
[76]	+				+				+			
[76]	+				+				+			
[83]	+		+	+		+			+		+	+
[86]	+					+			+		+	
[110]							+	+				
[111]							+		+	+		
[107]							+	+				
[108]	+				+				+		+	
[120]	+		+	+	+						+	+
[120]	+								+			
[122]							+		+	+		
[125]	+	+			+				+		+	
[137]	+		+	+	+						+	+
[151]	+		+	+		+			+		+	+
[149]	+		+	+		+			+		+	+
[180]	+		+	+	+						+	+
[193]	+	+	+	+		+			+		+	
[206]	+				+				+		+	+
[210]	+			+	+				+		+	+
[214]	+					+			+		+	
Total	25	4	11	13	14	9	4	2	22	2	21	11

Table 2.2: Sets of labels, where at least one label is used no more than in one corpus. Each row corresponds to a single corpus

References	Labels used
[2]	Certainty, mixed (in between certainty and uncertainty), neutral, uncertainty
[17, 20, 147, 102, 106, 194]	Angry, boredom, emphatic, helpless, irritated, joyful, motherese, neutral, other, reprimanding, surprised, touchy
[19, 18, 21]	Anger, emphathy, motherese, neutral
[26]	Anger, non-anger
[30]	Anger, boredom, doubt, neutral
[115]	Anger, anxiousness, disappointment, displeased, fear, happiness, impatience, irony, neutral, pleased, questioning, sadness, satisfaction, stress, surprise, unsatisfied, various labels, weariness (free labels)
[44]	Anger, fear, happiness, neutral, sadness, surprise, Undecided + 5 intensity levels
[51]	Anger, fear, hurt, relief, neutral, other positive, sadness, surprise + 20 more granulated labels
[53]	Boredom, confusion, delight, flow, frustration, neutral, surprise
[89, 88]	Aggressiveness, happiness, neutral, sadness, sensibility
[98]	Embarrassment (yes/no), pleasure (yes/no), affinity (familiar, neutral or tense)
[97]	Laughter, non-laughter
[100]	Non-sleepiness, sleepiness
[122]	Empathy, negative, neutral
[124]	Neutral, neutral-low, neutral-stressed, stressed
[160]	Approval, attention, neutral, prohibition, soothing
[162, 160]	Approval, attention bid, prohibition + strength on 1—5 scale
[169]	Anger, disgust, fear, happiness, nervousness/excitement, neutral, sadness, surprise
[184]	Amusement, anger, boredom, disgust, excitement, fear, frustration, happiness, malicious delight, relief, surprise, wonderment
[189]	Anger, excuse, fear, neutral, satisfaction + 20 labels for minor emotion
[190]	Anger, fear, neutral, relief, sadness
[194]	Anger, joy, pleasure, sadness
[208]	Anger, annoyance, happiness, neutral, sadness

label sets, where at least one label is used just in one corpus. More surveys on corpora could be found in some sources devoted to emotion recognition in speech [40, 52, 188].

Many corpora use task-dependent labels. For example, to detect anger only two labels are needed—anger and absence of anger [26], call center applications need to classify mostly negative emotions [30] and training systems have to detect uncertainty [2]. But the overwhelming majority of researchers use six basic emotions and the neutral state (when no emotions are being exhibited) with slight variations. Some researchers propose labelling each instance not with one, but with two labels. The second label could be used in cases when two emotions are being expressed simultaneously to denote the minor one [52, 189] or to specify emotion more precisely [39, 51]. For example, if the first label is *Negative*, the second label provides a choice between *Annoyance*, *Anger* and *Disappointment*. In some cases the raters are required not only to pick up the emotion category from the list, but also to denote its intensity [44, 162, 160].

Sometimes the previously selected sets of labels perform in an unexpected way. For example, in one of the corpora labels *Eureka* had been considered too vague and was replaced by *Delight* and *Surprise* in the process of actual labelling [53]. There also might be cases when some emotion is being reported in too few cases, it has happened in the EFN corpus with *Disgust*. As a result, instances labelled by this emotion were discarded [44].

There might be also different approaches to just offering a list of labels and asking to select from them. For instance, in some work [103] a special tree was constructed to help labellers (see Figure 2.1) raters go through it answering the questions and finally they get to the leaf that contains the label that has to be reported.

Consensus labelling

When multiple raters are asked to label an instance with discrete categories, they often will not all choose the same label. It has been stated that agreement on one common label normally is possible only in a few cases [171]. The level of agreement between labellers can be measured in several ways and this is the topic of the next subsection. Here we will

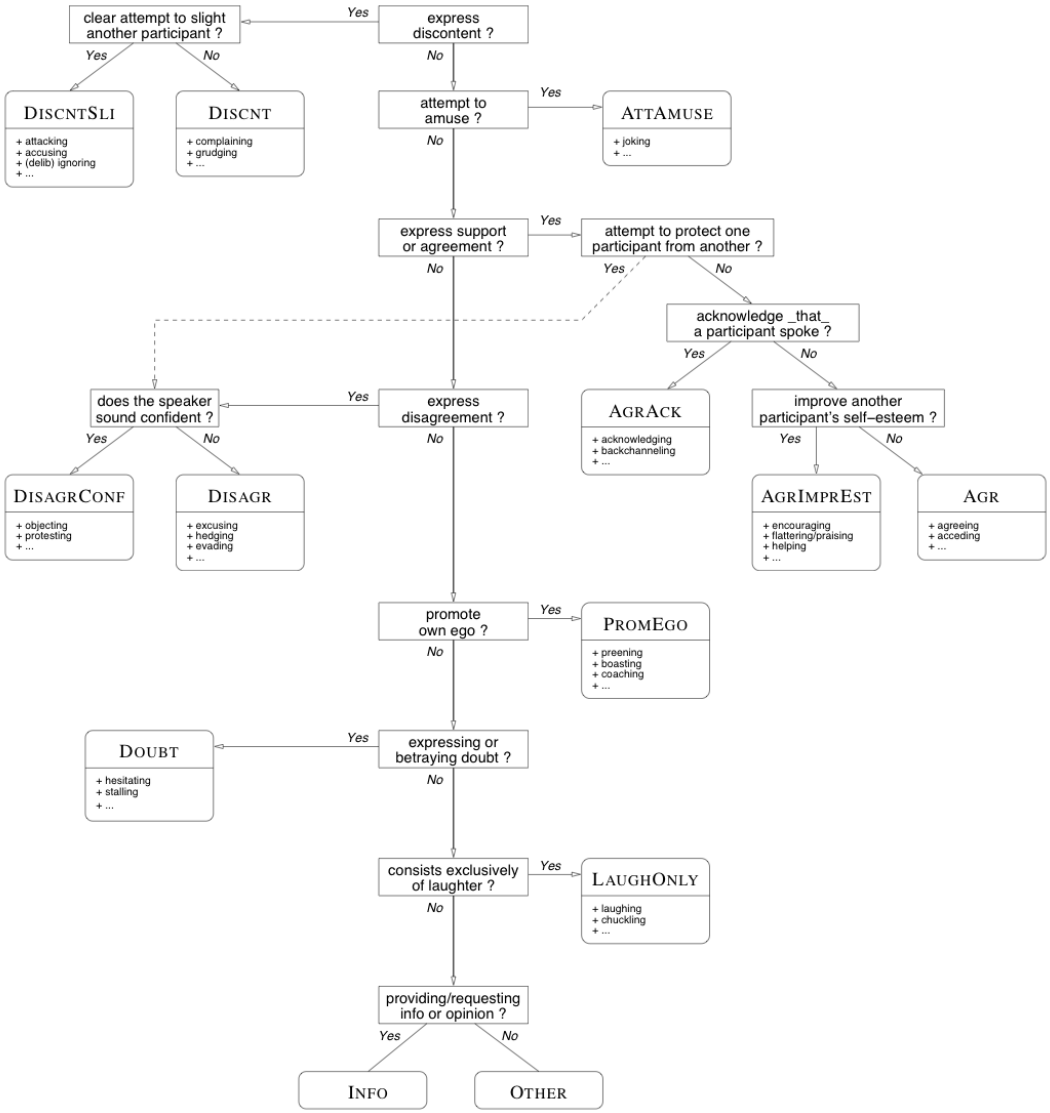


Figure 2.1: Decision tree constructed to help raters to label emotional instances [103]

concentrate on approaches that either choose a single label for the instance, or discard it as being too ambiguous.

The simplest approach is the majority voting—the label that has gathered the biggest number of "votes" from raters is assigned to the instance [189]. It does not work well in the case when all votes are almost equally divided across a few labels, and the probability of such an event increases as the number of raters decreases. For example, if there are only three labellers, almost equal count of votes often will be assigned to two or three categories (each getting just a one voice). Majority voting always will pick just one of them, even if others are close in the matter of votes. That is why the most part of state-of-the-art

research uses more sophisticated ways of assigning a single label when raters disagree that are described below.

Each instance in corpus BabyEars [162, 160] is labelled not only with emotion, but also with its strength, measured on a scale from one to five. Only those instances where at least five out of seven labellers agree and where the strength is above 2.5 are proposed as valid, others were discarded. The same approach was used in corpora EFN [44] and AIBO [20, 17, 147, 102, 194, 106] (but in the latter case there were no strengths associated with instances). Some researchers state even more strict criteria—agreement of at least 80% of experts [125] or even full agreement, i.e. only those cases where all raters choose the same label [98].

There is no agreement on a standard approach to consensus labelling in state-of-the-art research. Though we would like to point out that such a choice is always a trade-off between the number of instances in corpus (and how well any emotion is represented in it) and the general quality of corpus. For example, majority voting would not reduce the number of instances much, but then the corpus would contain ambiguous instances that could make classification more difficult. On the contrary, requiring 100% agreement would reduce the number of samples (and it could pose problems during the training process), but all of them will be quite good representatives of emotions.

Agreement measures

Some research reports that by excluding ambiguous cases from a training set (i.e. those, on which labellers disagree to a high extent) it is possible to raise the rate of correct classification dramatically, even for 30% [19]. Sometimes such exclusion could lead to a very small amount of data where labellers agree to a certain extent. This, in turn, could also lead to the decreased performance of the classifier, as well as exclusion described above. That is why a need to investigate where exactly the disagreement occurs and what can be done to make it less significant (e.g. change labelling system) exists.

Many measures from traditional statistics are not suited for the measurement of disagreement, because they can not measure the agreement directly or do it precisely enough

[9]. For instance, the percentage of agreements (the percentage of judgments on which raters agree when labelling the same data independently) does not take into consideration an agreement that can occur just by chance. As a result, a labelling schema with fewer choices would have a higher value of this measure than the one with more options not because the disagreement is lower, but because the number of choices is smaller. The inappropriateness of the correlation coefficient could be illustrated with a following example. Let the labels of classes be numbers from 1 to 10 and the number of instances to be classified is five. If two raters label the instances identically, for example, both produce vectors $(1, 2, 3, 4, 5)$, the correlation coefficient between them is one, so in this particular case the high value of the correlation coefficient denotes the high level of agreement. But if the labels assigned to the instances by raters are $(1, 2, 3, 4, 5)$ and $(2, 4, 6, 8, 10)$, the correlation coefficient will also be equal to one, though we have a disagreement. That is why different measures of consensus should be used.

One of the most widely used measures of agreement in emotion recognition is the κ-statistic [37]. It works in the same way as a percentage of agreements, but it takes into consideration the agreement that arises just by chance. This statistic is calculated in the following way:

$$\kappa = \frac{p_o - p_c}{1 - p_c}, \tag{2.1}$$

where p_o is the proportion of samples where agreement between labellers is seen and p_c is proportion of expected agreement "by chance", i.e. the agreement that was expected if experts would perform random labelling. Both p_o and p_c can take values between 0 and 1, the denominator shows what is the maximum non-accidental agreement is possible and the value in the numerator is the actual proportion of non-accidental agreement. The closer the κ to one, the better the agreement. The main deficiency of this statistic is that there is no value above which the agreement can be considered as significant, though some research states that values above 0.75 [170] or above 0.80 [32] express good agreement; below 0.40 [170] or below 0.67 [32] a bad one.

In fact, a lot of research makes use of the κ-statistic for emotion recognition [26, 30,

111, 107], but only a small part of labelled corpora expose values above 0.65 [26, 122]. In some research this statistic is used to discover which emotions are easier and more difficult to be recognized by raters [53], in some works confusion matrices are used for the measuring disagreement between raters [89, 88].

Formulas to calculate p_c can be found in research devoted to agreement measures [9]. A version of κ-statistic called weighted κ could take into account the severity of disagreements—for example, disagreement when one labeller states that utterance depicts "cold anger" while other insists on "hot anger" is less significant than if they are arguing on is it happy or angry [170].

Some research proposes that the κ measure can be inaccurate and offers to use Krippendorff's α-statistic [41]. It differs from κ in that it takes into consideration the frequency with which labels are used while calculating the level of agreement by chance. For example, α for the case of two raters and two possible labels is calculated as

$$\alpha = 1 - (n-1)\,\frac{o_{01}}{n_0 \cdot n_1}, \tag{2.2}$$

where o_{01} is the number of disagreements between two raters, n_0 (n_1) is the number of times label 0 (1) has been used and n is the total number of examples rated. The generalization of this criterion for bigger number of raters and classes can be found in the book by Krippendorff [101].

2.2.3 Emotional dimensions

This subsection describes how continuous dimensions are used in the task of emotion recognition from speech. We describe how dimensions are chosen, as well as how the output of many raters could be combined to produce a single label for the instance.

Choosing the dimensions

One of the most important deficiencies of discrete labeling approach that it cannot represent emotion precisely enough—emotions in daily speech are usually weak, moderately strong and mixed rather than pure [39]. It means that raters might have serious difficul-

Figure 2.2: Activation-evaluation model [170]

ties while assigning labels to instances—is some utterance angry or is it closer to neutral? One more problem is that the meaning of the label could be not the same for different people [186]. That is why the usage of special dimension models is proposed for labelling in state-of-the-art works in emotion recognition—many examples with references will be presented in this subsection. In such case not any emotion has an exact name, but is represented as a point in a coordinate system.

One possible model consists of two dimensions that are evaluation (also called valence or pleasure [145]) and activation (also called arousal or activity [145]). Evaluation [52] measures how positive or negative the emotion is—for example, happiness is very positive and sadness is very negative. Activation [52] describes the level of activity on a scale of passive to active—angriness is an example of an active emotion whereas tiredness could be considered passive. An example of how some discrete emotions could be approximately mapped to the activation-evaluation model is shown on Figure 2.2. There were some

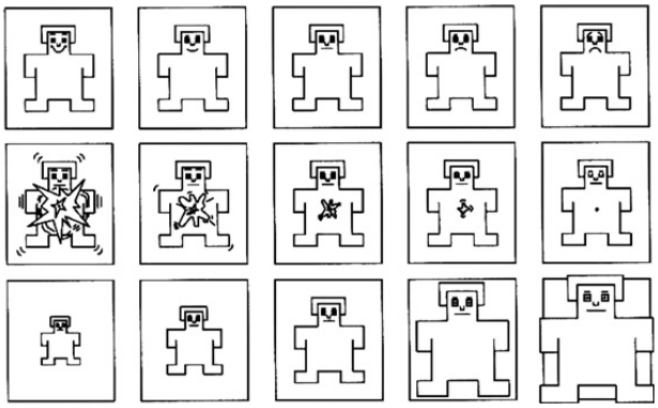


Figure 2.3: Self-assessment manikins, used for rating Vera am Mittag corpus: upper row presents evaluation, middle row evaluation and lower row dominance [72]

attempts to validate it by performing a mapping of discrete emotions to two dimensional plane based on their acoustical features, where two axes received could be interpreted as activation and evaluation [183]. In fact, mapping from categories to dimensions is a simple task, but inverse process is not possible [145].

Concerning emotion recognition in speech, the amount of activation is highly correlated with pitch, but the same characteristic for valence has not been found yet [119]. Some research points out that this dimension is better modelled by lexical than acoustic features [184]. On the contrary, discrete emotions do not have distinct "signatures" in any features at all [119].

One problem with this model is that some emotional states can not be separated from others. One of the most frequently used examples is anger and fear—both emotions have high evaluation and activation levels. Because it is impossible to separate these two emotions, people often prefer a categorical approach [51]. But in that particular case adding one more dimension, called dominance (commonly used synonyms are potency and power [145]), will help—this approach is used very widely in state-of-the-art research [70, 71, 73, 72]. Though some authors argue that adding more dimensions could allow many additional states to be discriminated [39], and we are unaware of any research that uses more than three dimensions for performing the labelling.

There is one more challenge associated with dimensional models—raters have to be trained to understand scales. One of techniques that may avoid this is called self-assessment

manikins, used for rating the Vera am Mittag Corpus [69, 71, 73, 72]. rater is presented not with just a scale where he has to pick a point, but with five pictures for each dimension (Figure 2.3) and he has to choose one which depicts the expressed emotion best.

Independent of the rating scale used, the final result of labelling is always the same: each instance has multiple ratings coming from different raters. The next subsection describes how they are combined to produce a single rating for each speech recording.

Consensus labelling

One of the most evident ways to combine output from several raters when emotional dimensions have been used for labelling is to take a mean value [72] or a median [186] of all ratings for each instance for each of the dimensions separately. Such an approach is used in nearly all research concerning emotional speech recognition. However, this approach does not take into account the fact that each rater can perform differently and provide unreliable rating systematically.

One of a few ways to handle the problem of rating aggregation different than simple averaging is suggested by [69]. It is based on using correlation coefficients and works in the following way. Let us denote a rating for the i-th dimension ($i = (1, 2, ..., I)$, where I is a number of dimensions) from the j-th labeller ($j = (1, 2, ..., N)$, where N is a number of raters) for the k-th instance ($k = (1, 2, ..., K)$, where K is a number of instances) as $x_j^{(i)}(k)$. Then the maximum likelihood estimation for the rating for i-th dimension for k-th instance is a mean that is calculated in the following way:

$$x_k^{MLE,(i)} = \frac{1}{N} \sum_{j=1}^{N} x_j^{(i)}(k) \tag{2.3}$$

The sequence of ratings for all instances by j-th expert for the dimension i could be expressed in the following way:

$$\mathbf{x}_j^{(i)} = \left(x_j^{(i)}(1), x_j^{(i)}(2), ..., x_j^{(i)}(K) \right) \tag{2.4}$$

The sequence of maximum likelihood estimates for all instances can be written as

$$\mathbf{x}^{MLE,(i)} = \left(x_1^{MLE,(i)}, x_2^{MLE,(i)}, ..., x_K^{MLE,(i)}\right) \tag{2.5}$$

Then the correlation coefficients $r_j^{(i)}(j = 1..N)$ between expression (2.4) and expression (2.5) can be calculated. If this value is smaller than zero, it is considered to be equal to zero. The smaller it is, the less reliable the j-th labeller is. The weighted estimator for the i-th dimension of the k-th instance then can be expressed as

$$x_k^{EWE,(i)} = \frac{1}{\sum_{j=1}^{N} r_j^{(i)}} \sum_{j=1}^{N} r_j^{(i)} x_j^{(i)}(k) \tag{2.6}$$

Unfortunately, this measure does not take into account that the correlation coefficient is not suitable for measuring agreement, as was previously described in subsection 2.2.2, although this is not mentioned in the research using this measure [69].

The agreement between labellers can be measured as the standard deviation of labels of each dimension separately [69, 72]. A more sophisticated way of doing it is calculating inter quartile range and comparing it with some predefined value [186]. The instances for which the range is bigger than the predefined value are considered too ambiguous to be included to a training set.

2.3 Conclusions

This section briefly went through the process of supervised machine learning, using emotion recognition from speech as an illustrative example. Two main ways of labelling emotion, using discrete categories and continuous dimensions, were described in detail. Currently there is no consensus in which model—discrete or dimensional—is more effective in emotion recognition tasks. Some research [50, 184] offers both approaches for the same datasets, but no conclusions on which should be preferred are drawn. Though it should be noted that the discrete emotion approach works better for acted than for elicited or natural speech, where dimensions can perform better.

It can be seen that most researchers use multiple people to rate emotional speech in

such a way that a resulting rating for a recording is a combination of ratings submitted by multiple raters. This makes the rating of emotional speech a natural crowdsourcing task. In the next chapter we describe crowdsourcing in more detail.

Chapter 3

Crowdsourcing

Crowdsourcing, which by the original definition is "the act of taking a job traditionally performed by a designated agent and outsourcing it to an undefined, generally large group of people in the form of an open call" [82] is used to solve a lot of different problems including conversion of paper documents to electronic form[1], proofreading and editing[2] and even the design of t-shirts[3]. The application area we are interested in is rating of datasets for supervised machine learning, where crowdsourcing is used very widely not only in emotion recognition from speech [11, 72], which is the motivating example of this thesis, but also in machine translation [8, 31], sentiment analysis [22, 84], image annotation/classification [1, 123, 168, 198], natural language processing [55, 165] and many other application areas of supervised machine learning. In general, crowdsourcing facilitates getting ratings quickly and cheaply—for instance, Sorokin and Forsyth [168] collected 3,861 ratings with a speed of 300 annotations per hour for just 59 USD compared to as much as 1,000 USD or 5,000 USD, if the rating was performed by experts. Such savings are possible in many areas, and a large number of rating tasks are always available on Amazon Mechanical Turk (AMT)[4]—a special marketplace, where anybody can sign up and solve tasks, usually, for micropayments of a few cents.

A typical crowdsourcing tasks involves posting a large number of tasks online and

[1] http://microtask.com/
[2] http://www.serv.io/
[3] http://www.threadless.com/
[4] http://www.mturk.com

inviting people to complete them. Usually, multiple answers are collected for each task, which are aggregated in some way to produce a single answer for every task. In the context of rating corpora for supervised machine learning, a task usually represents rating a single training instance. The simplest way to aggregate ratings coming from the raters is to let them rate as many instances as they want, presenting instances to the raters one by one in a random order and letting a rater to rate an instance only once. When N ratings for each rating is gathered, the process stops and predictions are calculated for each instance as average of ratings submitted for that instance.

One of the main challenges in crowdsourcing is the existence of noisy raters. They would submit inaccurate ratings either because they lack skills, or because they want to get the reward without investing a real effort. If ratings are aggregated as described above, such noisy raters can have a negative impact on the accuracy of the resulting predictions. In order to reduce the effect of noisy ratings, different quality control techniques can be used. Most of the techniques can be divided into two groups, which we call *static* and *dynamic*. Static approaches proceed as described above, but apply an expectation maximisation algorithm at the end of the rating process instead of just averaging. Such algorithm calculates rater reliabilities and uses them as weights when calculating the predictions in such a way that ratings coming from noisy raters get low weights and do not affect the resulting predictions. A dynamic approach works in a different way: instead of estimating rater reliability only once, statically, at the end of the process, it tracks rater reliability as raters rate instances. Only raters who deemed to be reliable to date are allowed to rate, therefore, ratings from noisy raters are not even collected. Developing such a dynamic approach, that can be used in real-life crowdsourcing scenario, is the main focus of this thesis.

Despite a large variety of both static and dynamic techniques, the state-of-the-art research usually does not address the question of the choice between these two kinds of quality control. The purpose of this chapter is to explore a variety of crowdsourcing tasks and to develop recommendations on when the dynamic approaches are especially beneficial and should be preferred to static ones in order to reduce costs associated with the rating

process. In order to make such recommendations, we developed a categorisation of rating tasks and illustrated it with some examples.

In this thesis we propose to represent the problem of dynamic estimation of rater reliability as a multi-armed bandit problem. A multi-armed bandit is a mathematical abstraction representing the task at hand as a multi-armed gambling machine where some arms are better than others. When an arm is pulled, it produces a numerical reward, and good arms tend to produce relatively high rewards compared to other arms. The goal is to find the best arms as quickly as possible by pulling different arms in a certain way that helps to discover the best arms as quickly as possible. The rating process can be represented as a multi-armed bandit problem: there is a need to find the most reliable raters, those who provide the best ratings, in the shortest time possible. For the task of rating supervised learning training data, each available rater corresponds to an arm. At each moment of time we can choose rater(s) in the full rater population from whom to solicit a rating for a training instance. Asking a rater to provide a rating for an instance is equivalent to pulling an arm. The reward received after selecting a rater (or pulling an arm) is proportional to the accuracy of the rating received. In this chapter we briefly describe the main concepts of multi-armed bandits, as well as cover how these techniques have been previously applied to crowdsourcing tasks.

This chapter is structured as follows. Section 3.1 covers the related work in defining crowdsourcing. Section 3.2 explores a wide range of crowdsourcing tasks by looking into different categorisations and taxonomies developed to date. Section 3.3 is devoted to using crowdsourcing in rating corpora for supervised machine learning. Section 3.4 analyses dimensions from the existing categorisations and proposes some new dimensions, which are especially vital for rating tasks. In Section 3.5 we briefly describe multi-armed bandits. Section 3.6 contains the review concerning the use of multi-armed bandits in crowdsourcing tasks. Section 3.7 concludes the chapter.

3.1 Defining crowdsourcing

According to Estellés-Arolas and Gonzalez-Ladron-de Guevara [59] the limits of crowdsourcing currently are blurred due to the diversity of practices, which are used in it. Likewise, Rouse [141] states that current definitions of crowdsourcing are "wooly and use claims, based largely on the basis of anecdote rather than systematic study". Such situation can be explained by the fact that one of the first extensive sources on crowdsourcing [82] was composed with the purpose of exploring the possibilities of collaboration between individuals in different business contexts, rather than coming up with a precisely defined scientific term. Howe [82] tries to make parallels between completely different application areas such as banks of pictures, prediction markets and even a movie, which plot was composed by a crowd. Such diversity opens the possibilities of further refining the definition of crowdsourcing in many different ways. Indeed, Estellés-Arolas and Gonzalez-Ladron-de Guevara [59] counted forty different definitions of crowdsourcing, some of which even contradicted each other. However, they were able to come up with the integrating definition: "crowdsourcing is a type of participative online activity in which an individual, an institution, a non-profit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task."

Crowdsourcing is also very widely used to collect ratings for data to be used in supervised machine learning. A usual scenario for this is to use tens and hundreds of workers with undefined skills or background, recruited through the Web [61, 84, 165]. At the same time, there is some work, which uses relatively small crowds, where workers are expected to have certain special skills and therefore, not hired through crowdsourcing platforms such as AMT. For instance, Raykar et al. [135] describe the task of medical imaging, where four or five radiologists are rating an X-ray image, by saying if the lesion is benign or malignant. Likewise, Smyth et al. [163] used five planetary geologists to find volcanoes on the photos of Venus surface, while Batliner et al. [20] reported five advanced students of linguistics rating emotional speech corpus. None of those and similar works mention

the term *crowdsourcing* explicitly, however, many of such tasks involving multiple raters can be considered crowdsourcing tasks according to the definition by Estellés-Arolas and Gonzalez-Ladron-de Guevara [59] given above. There are three main differences between the "standard" scenario and multiple rater tasks we mentioned above: the number of workers, the requirements for workers and the way workers are hired.

No constraints on a number of workers is present in the definition by Estellés-Arolas and Gonzalez-Ladron-de Guevara [59]—it can be as small or as large as required by a specific task. Estellés-Arolas and Gonzalez-Ladron-de Guevara [59] claim that sometimes the size of the crowd might be limited by "those within a company [or] those that deal with confidential information" as well as that certain tasks need a specially educated crowd. According to them, each crowdsourcing initiative requires a different number of workers and varying requirements considering knowledge and skills they should possess. The task itself can also change the definition of the "open call". According to Estellés-Arolas and Gonzalez-Ladron-de Guevara [59], some researchers believe that a truly open call should not be limited to experts of preselected candidates, however, there is no full consensus about it among the research community. Whitla [199] lists three different types of the open call:

1. A true open call where any given interested party can participate.
2. A call limited to a community with specific knowledge and expertise.
3. A combination of both, where an open call is made, but those who can participate are controlled.

Therefore, according to the definition by Estellés-Arolas and Gonzalez-Ladron-de Guevara [59], initiatives, where a relatively small number of experts is used, can be regarded as crowdsourcing as well, despite the fact that authors do not always use this term explicitly.

However, the problem of varying definitions is not the only problem in defining crowdsourcing. Crowdsourcing is often confused with related, but different concepts such as *human computation*, *social computing* and *collective intelligence*. The main goal of the

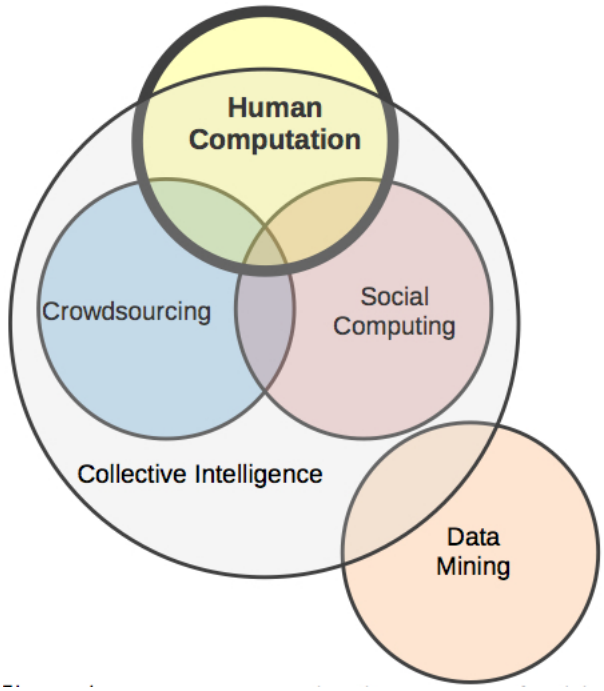


Figure 3.1: Crowdsourcing and related terms [133]

work by Quinn and Bederson [133] is to draw distinctions between those terms as well as to show where they overlap (Figure 3.1).

Collective intelligence is the most general concept, and it studies "groups of individuals doing things collectively that seem intelligent". Evidently, it is a superset of crowdsourcing. It also overlaps with human computation, which by original definition is "a paradigm for utilizing human processing power to solve problems that computers cannot yet solve" [104]. One of the historically first examples of human computation is the ESP game[5], the main purpose of which is to use human effort to tag images. The rules are simple—an image is shown to two players, and each of them has to guess which keywords the other used to describe the image. Indeed, there is an element of crowdsourcing in the ESP game, because the tags are the result of a collaboration between number of people from the crowd. However, the process is coordinated by presenting images, calculating scores and ensuring that participants follow the rules, which makes this game a human computing task. Another example is the reCAPTCHA[6] tool, which presents to users fragments of scanned paper books or articles and asks them to decipher them. Both reCAPTCHA

[5]http://www.gwap.com/gwap/gamesPreview/espgame/
[6]http://www.google.com/recaptcha

and ESP game do not involve any monetary costs, however, they still would be considered human computation tasks even if raters had to be paid. It can be seen, that both make use of a group of people, thus, they represent the overlap of human computation and collective intelligence. However, human computation can also utilise just one person, for instance, performing decyphering of CAPTCHAs by one hired rater. Such human computation tasks lie outside collective intelligence.

There is an overlap between human computation and crowdsourcing, for instance, Law [104] mentions the AMT as an example of human computation system, because it explicitly controls how tasks are executed. For instance, AMT can filter perspective participants by some criteria, which are pre-defined by the author of the task (for instance, only people from a certain country are eligible). Also, AMT performs the assignment of tasks as well as taking care of the details as to how rewards are transmitted to workers. It also monitors the performance of workers on previous tasks, expressed in a percentage of answers that were accepted by owners of these tasks. This performance can be used as one of the pre-selection criteria. In contrast, crowdsourcing platforms like Threadless.com do not have such sophisticated control functions. All it does is gathering designs from users, posting them for voting, and then presenting the top scoring ones to a board of human experts, who chose the designs to be rewarded and produced. Such tasks are representatives of "pure" crowdsourcing tasks, i.e. tasks, which belong to crowdsourcing, but not to human computation.

Another concept mentioned by Quinn and Bederson [133] is social computing, the technology facilitating relatively natural human behaviour. The main difference between human computation/crowdsourcing and social computing is that usually the task in social computing is not to solve some problem by finding an answer, but, for instance, to aggregate knowledge (Wikipedia) or to facilitate interaction online (YouTube). The authors do not provide any example where social computing overlaps with human computation and crowdsourcing which might indicate some potentially interesting research areas. It should be mentioned, that the same is true for many other cases as well, for instance, using the definition of crowdsourcing by Estellés-Arolas and Gonzalez-Ladron-de Guevara [59], it is

difficult to imagine tasks, which would involve human computation, be a part of collective intelligence, but at the same time would not be crowdsourcing tasks.

However, even if only relatively "pure" crowdsourcing tasks are considered, there is a huge variety among them with respect to how the crowd is handled, how the worker results are aggregated and so on. Numerous approaches to classifying crowdsourcing tasks were suggested in recent years. Section 3.2 is devoted to describing main of them.

3.2 Taxonomies of crowdsourcing

A survey of such taxonomies is given in Table 3.1. As can be seen, the purpose of the most of them is to understand the definition of crowdsourcing better. Typically researchers look at a very broad range of crowdsourcing tasks, sometimes covering concepts such as open source software in their classifications as well. That can explain a big variety of ways of looking at the categorisation, however, there are some dimensions, which are quite common. In order to analyse dimensions, we divide them into three categories: ones, related to the participants; to the task at hand; and to the crowdsourcing process, i.e. how exactly participants are performing tasks and what happens to the results. The following subsections provide a more detailed description of these three groups.

3.2.1 Dimensions related to participants

The most widely used dimension of all is *motivation for the workers*, which is also is one of the most significant challenges in crowdsourcing [54]. Doan et al. [54] list five different ways participants can be recruited and retained:

1. Use authority to require workers to perform tasks.
2. Pay workers for their contribution.
3. Ask workers to volunteer.
4. The work is necessary to use a certain system (for instance, in order to access the information about plane tickets user has to solve a CAPTCHA, which is in fact a

fragment of text, which somebody wants to get recognised).

5. Piggyback on the user traces of a well-established system, such as building a spellchecker based on search queries entered by users (*piggyback vs. standalone* is also mentioned as a separate dimension by Doan et al. [54]).

Table 3.1: Overview of crowdsourcing taxonomies in literature, based on work by Geiger et al. [66], but significantly updated.

Reference	**Purpose of work**	**Dimension**
Corney et al. [38]	Foundation for identifying methodologies or analysis methods	Nature of the task: creation of something new, choice between existing options etc.
		Requirements for crowd
		Payment strategy: is there any payment and if there is, is it money or something else, such as an item
Doan et al. [54]	Global picture of crowdsourcing systems on the Web	Explicit or implicit work: are workers aware that they are actually working. An example of implicit work is reCAPTCHA or ESP game.
		Type of the task: evaluation of something, building an artefact, sharing some content etc.
		Degrees of freedom: is the task narrow and pre-defined or is it a creative task
		Motivation for the workers
		Aggregation of worker results
		Quality control policy
		Degree of automation: relatively small when combining ratings, much bigger when combining pieces of computer code, submitted by workers

Continued on next page

Table 3.1: Continued from previous page

Reference	Purpose of work	Dimension
		Participants role: solving a simple multiple-choice problem, contributing self-generated content etc.
		Piggyback vs standalone: is the crowdsourcing system built in some other system or not
Erickson [57]	Framework for thinking about and designing systems that support crowdsourcing, human and social computation	Do workers work at the same time or not
		Do workers work at the same place or not
Geiger et al. [66]	Systematization of processes which are used in crowdsourcing	Requirements for crowd
		Degree of collaboration between workers
		Aggregation of worker results
		Payment strategy
Malone et al. [116]	Identifying the building blocks of collective intelligence approaches	Nature of the task
		Requirements for crowd
		Motivation for the workers
		Design of the task: do workers provide answers, which are combined afterwards; do workers compete with each other etc.
Piller et al. [130]	Analysing strategies for customer participation in open innovation	Stage in the innovation process: is the output from workers used at the stage of idea generation, concept development, product design, testing etc.
		Degree of collaboration between workers
		Degrees of freedom
Quinn and Bederson [132]	Classification and comparison of distributed human computation systems and ideas	Motivation for the workers
		Quality control policy
		Aggregation of worker results
		Human skill, which is required for the task
		Participation time
		Cognitive load

Continued on next page

Table 3.1: Continued from previous page

Reference	Purpose of work	Dimension
Quinn and Bederson [133]	A common understanding of human computation systems	Motivation for the workers
		Quality control policy
		Aggregation of worker results
		Human skill, which is required for the task
		The way how workers and requesters interact
		Task-request cardinality: how many people are allowed to solve one task and how many people are required to solve a single task
Rouse [141]	Clarifying the definition of crowdsourcing	Requirements for crowd
		Principal beneficiary
		Motivation for the workers
Schenk and Guittard [142]	Understanding crowdsourcing from management perspective	Aggregation of worker results
		Complexity of the task
Wightman [200]	Considerations for system design, identifying directions for further research	Competition between workers
		Motivation for the workers
Zwass [217]	Taxonomic framework as prerequisite for theory building in co-creation research	Explicit or implicit work
		Requirements for crowd
		Motivation for the workers
		Management of the process: are workers autonomous or do they have to obey certain rules, how strict these rules are etc.
		Task characteristics: time frame for completion of the task(s), intellective demands, effort intensity etc.
		Aggregation of worker results
		Principal beneficiary

The approach taken by Malone et al. [116] also acknowledges that workers can be paid for their activity, but at the same time it includes two other ways—*love* (workers enjoy carrying out the tasks and do not expect a monetary gain) and *glory* (workers seek to be recognised by peers for the contribution). Some authors [133, 132] propose a combination

of the approaches by Doan et al. [54] and Malone et al. [116]. Zwass [217] used the above mentioned criteria, but he also argues that the workers might be motivated by the use of the result of the task (such as aggregated Web bookmarks), learning new skills (mentioned by Rouse [141] as well), competition (also mentioned by Wightman [200]) or forming a personal relationship. Some researchers [54, 217] also draw a distinction between *explicit or implicit work*, i.e. whether workers are aware that they are actually doing some work, which is somewhat related to the question of worker motivation, but looks at it in a more general sense.

One more dimension, which is relevant to motivation, is the *payment strategy*. Corney et al. [38] suggest that all the tasks can be voluntary (no payment is involved), paid a flat rate for the accepted work (typical crowdsourcing conditions, when some payment is given for each task performed), or rewarded with a prize (in such conditions only one winner is selected typically). Geiger et al. [66] propose a different three-class classification: (i) no payment involved, (ii) everybody is paid, even if the solution did not contribute significantly to the final result, (iii) only valuable contributions are paid for.

Many researchers also consider *requirements for crowd* as a way of distinguishing crowdsourcing tasks. Malone et al. [116] proposes a simple distinction between tasks on this dimension: they are either suitable for being solved by "crowd", a large group of individuals, or should be solved by a smaller pre-selected group. Corney et al. [38] uses the same categories with adding one in the middle—tasks, which require a certain, however, not unique skill (for instance, translation tasks). Geiger et al. [66] makes a distinction between qualification- and content-based pre-selection. Context-specific reasons include demographic requirements such as being of a certain age, living in a certain country, or being a customer/employee of a certain organisation. In some cases, no preselection is required, in some both qualification- and content-based pre-selection are used. The classification by Zwass [217] combines difficulty and context together, separating tasks by the group of potential contributors into "world" (almost any individual can be a participant), pre-qualified individuals (the individual should comply with some pre-condition, such as had been staying in a certain hotel) and contributors having a particular skill or belonging

to a certain community.

Doan et al. [54] propose four categories of *participants role* (each participant can have multiple roles):

1. Slaves—workers are providing an answer to the task, which has been decomposed via divide-and-conquer approach.

2. Perspective providers—workers are providing different perspectives and opinions, for example, reviewing books.

3. Content providers—workers contribute some self-generated content such as videos.

4. Component providers—workers function as components in a target artifact, for instance, in open-source software development.

Quinn and Bederson [132, 133] suggest that crowdsourcing initiatives can also be classified by *human skill, which is required for the task*, however, they do not propose any taxonomy of such skills.

A relatively unusual view of the classification of participants is suggested by Erickson [57]. His position paper suggests to take into account the fact if *participants are located in the same place or not* and if the *participants are doing work at the same time or not.* However, the author admits that the goal of this categorisation is more to provoke a discussion than to be used for some practical purpose immediately.

3.2.2 Dimensions related to tasks

Some researchers consider that *degrees of freedom* of the task should be taken into consideration while classifying crowdsourcing initiatives. Doan et al. [54] and Piller et al. [130] do not provide any defined categories, but note that there is a broad range from the tasks where workers are supposed to carry out a simple task, such as answer multiple choice question, to creative tasks with a solution, which is not known even approximately. Similarly, Schenk and Guittard [142] categorise tasks on the *complexity of task* dimension into creative, simple and complex knowledge-intensive ones. The same idea is proposed

by Malone et al. [116]: they call such dimension *nature of the task* and suggest two categories: tasks, which involve creation of something new (for instance, a design of a t-shirt or a new algorithm to solve some problem), and tasks, where workers have to make decision, choosing one of possible alternatives (such as selecting a t-shirt, which will be put to production). Corney et al. [38] also have the category of "creating something new", but they further subdivide "make decision tasks" proposed by Malone et al. [116] by drawing a distinction between tasks, where a survey is involved (choosing the best t-shirt or different social science surveys), and organisation tasks, which involve the organisation of information (for instance, tagging images or submitting translations). Additionally, Malone et al. [116] introduced a *design of the task* category, which has the following kinds of tasks:

1. Collection—all results are gathered independently of each other, and the final results is the whole collection, such as videos on YouTube. Although each contribution might be worthwhile by itself, the collection of all contributions has a much bigger value.

2. Contest—is a subtype of collection, where one or several items of it are designated as the best entries and receive some reward.

3. Collaboration—workers are creating some artifact together (open source software).

4. Group decision—the results of all workers are aggregated via voting, averaging etc.

5. Individual decision—the individual results of workers are worthwhile by themselves and are not aggregated (for instance, free-text reviews of products on Amazon.com).

The idea of Doan et al. [54] is somewhat similar—they suggest to divide the tasks by the *type of the task*—creating a new artifact, evaluation of something, sharing content etc.

Another angle to look at the tasks is to consider the *cognitive load*, which is required in order to solve them. Quinn and Bederson [132] offer three categories on this dimension: (i) passive processing tasks, almost no attention is necessary, (ii) tasks requiring to solve a single-step problem, requiring domain knowledge or creative thinking and (iii) tasks,

where the worker is under time pressure, solves a complicated multi-step problem or suffers another significant cognitive load.

Other dimensions used in state-of-the-art research include *participation time* dimension [132], which denotes the minimal time, which is required to complete the task, and *task characteristics* [217]. However, the latter contains a lot of different factors, such as task complexity, time frame and other dimensions, which usually are viewed separately by other researchers.

3.2.3 Dimensions related to processes

Zwass [217] provide a very generic view on the *management of the process* of crowdsourcing. He lists many different schemas such as individual autonomy (no coordination of workers is carried out at all), using specific software code for the management of the process (for instance, version-control systems, which serve to organise open-source software projects) and bureaucracy (formal rules and responsibilities). Each of these ways of governance contains multiple aspects, such as the way how workers collaborate with each other, how their results are aggregated, what quality control looks like and many others.

One of quite widely used dimensions involving processes is the *degree of collaboration between workers*. The work by Piller et al. [130], which is devoted to the way customers can contribute to innovative activities of some company, lists only two possible modes: there is a collaboration between customers and there is not. Geiger et al. [66] consider a more general case and suggest that existing of collaboration can consist of workers viewing contributions of others, assessing or even modifying them.

Another widely used dimension is the way *aggregation of worker results* takes place. The simplest approach to categorise tasks on this dimension [66, 142] is to divide them into selective (solution of one worker is chosen and used as a resulting one) and integrative (results for each task coming from several workers are combined in some way). The discussion provided by Doan et al. [54] concentrates mostly on selective tasks, pointing out that this process can be relatively easy (averaging numerical outputs from workers) or much more complicated (integration of software source code fragments). Quinn and

Bederson [132] approach this category in a more technical way and suggest that different contributions can be aggregated using knowledge bases (each answer either adds a new fact or improve the quality of the existing one) or statistical methods (sometimes as simple as just averaging). They also consider a category of *grand search* tasks, where workers are asked to perform a search for some object, person or scientific phenomenon through photographs or videos. In this case, only a small portion of searches will lead to a meaningful conclusion, thus, it can be considered an integrative strategy. Another category proposed by Quinn and Bederson [132] contains *unit tasks*, where different results for the same tasks are independent of each other, thus, they do not need to be aggregated in any way. An example of such kind of task is submission of reviews for movies or books. Quinn and Bederson [132] refine their understanding of aggregation dimension in another paper [133] by adding tasks, answers for which are iteratively improved by giving the outputs of some workers to different workers and asking to improve them. They also mention the method of aggregation, where worker outputs are used to train a classifier using active learning as well as the opportunity to use genetic algorithms in order to combine the outputs of different workers. Zwass [217] also includes quality control in his classification and argues that competition, voting or moderators/auditors/facilitators can be used in order to come up with a final solution for the task (similar to integrative strategy).

Some researchers suggest to look at *quality control policy* separately, not in the context of the aggregation of results. According to Quinn and Bederson [132], the most widely used quality checking methods are:

1. Statistical filtering—for instance, filter out all responses that do not fit to a certain distribution.

2. Redundancy—multiple answers are gathered for each task, then the results are combined or a single answer (deemed to be the best one) is chosen as a final one.

3. Multilevel review—the answers, given by workers, are analysed by a different group of workers.

4. Expert review—a trusted expert reviews all answers.

5. Forced agreement—two or more contributors are working on the same task, and the answer is not accepted, until both workers agree on it. The ESP game is based on this principle.

6. Automatic check—in some contexts such as AI planning, solving the problem might be tricky, but checking if the answer is feasible is easy. For instance, it can be checked whether the constraints of the task are observed or whether the outcome is correct. Such checks can be used to discard invalid outputs.

7. Reputation system—in systems similar to AMT workers can engage in many tasks. Some statistics, such as the percentage of previously accepted answers, can be recorded and used in order to filter unreliable raters.

8. No quality control at all.

The work by Quinn and Bederson [133] uses the same list, but adds a few other options. One of them is the usage of economic models, where the monetary reward is paid according to a game-theoretic model, which reduces the incentive to cheat. Also, they suggest to use *ground truth seeding*, a mechanism, which assumes that we know the correct solutions to some of the tasks. If they are known, then they can be compared to the answers given by workers. Another way to control quality mentioned by Quinn and Bederson [133] is defensive task design, which makes cheating as time-consuming as genuine work. For instance, Kittur et al. [96] considered the rating of Wikipedia articles, and in order to prevent cheating, they asked raters to provide a detailed free-text explanation of their rating decision. The classification of quality control mechanisms proposed by Doan et al. [54] is much less technical than that mentioned above. They divide all such techniques into categories, which *defer* (by banning or "public shaming" of inaccurate raters), *detect* (by performing automated or manual inspections of workers' results) or *block* (for example, allow anyone to submit data, but only certain domain scientists to clean and merge this data into the central database).

Some work suggests making a distinction between crowdsourcing tasks based on who is the *principal beneficiary* of the final results. According to Rouse [141], all initiatives

similar to crowdsourcing can be divided into individualistic (where a single individual or a company is the principal beneficiary) and those where a broader community wins (such as open-source software projects). Zwass [217] take it further and offer a finer gradation of beneficiaries, however, it concentrates particularly on the innovation created by collaborative efforts:

1. The whole world (Wikipedia).
2. A community (users of a certain product, which is improved by collective efforts).
3. A single organisation.
4. The contributors and the sponsor (for instance, Amazon Mechanical Turk).

One more categorisation is based on the *task-request cardinality*, i.e. how many people are required to solve the task and how many tasks can be solved by each worker:

1. One-to-one—one worker performs a task alone. An example of this kind of cardinality is ChaCha application[7], where everybody can ask a question and get a single answer to it, provided by a worker.
2. Many-to-many—there are many tasks, each requiring multiple answers. Tasks, where a database of images has to be tagged (for instance, by denoting which season is depicted) belong to this category.
3. Many-to-one—a big number of people are trying to solve a single task (for instance, to solve a problem posted on Kaggle).
4. Few-to-one—a relatively small number of workers respond to one task (VizWiz service[8], where a blind person takes a photo of an object and a few workers identify the object, depicted on it).

Some less widely used categories include a distinction between tasks, which has a low *degree of automation* (such as combining software code), and which has much higher

[7] http://www.chacha.com/
[8] http://www.vizwiz.org/

one (combining answers from different workers by averaging them) [54]. Wightman [200] proposed to distinguish the task on the basis if there is a *competition between workers* (an example of the existence of the competition is InnoCentive platform, where each player competes for the prize). Quinn and Bederson [133] list four ways of how *workers and requesters interact with each other* ranging from simple ways, when the results are going directly to the requester, to more complicated schemas such as the requester accessing the database, which is composed by some algorithm using output from workers. Piller et al. [130] consider a highly domain-specific dimension—the *stage in the innovation process*, where crowdsourcing takes a place (it can be introduced early at the stage of product design or much later, when the testing begins).

As can be seen, there is a big variety of crowdsourcing tasks that can be categorised along many dimensions. In Section 3.3 we focus on one category of crowdsourcing tasks, namely, rating corpora to be used in supervised machine learning, which is the subject of this thesis.

3.3 Crowdsourced rating of corpora

Crowdsourcing is applied to different rating tasks, however, the reasons for using it are different. The classification of human computing tasks by Law and Von Ahn [105] underlines these differences, dividing rating tasks into those with *cultural truth* and those with *objective truth*.

In tasks with cultural truth the rating refers to the shared beliefs amongst the set of people that we ask, and usually involves some sort of perceptual judgement. The task of rating emotional speech recordings if a speaker is angry, neutral, excited etc. [20], is an example of this kind of task, since emotions are perceived differently by individuals.

In contrast to tasks with cultural truth, tasks with objective truth have an answer, which is external to human judgement. One of the simplest tasks of this kind is the *textual entailment task*, wherein a rater is presented with two sentences and given a binary choice of whether the second sentence can be inferred from the first [165]. This choice will

depend on the laws of logics, not on the human judgement or perception. In some tasks with objective truth it can be quite difficult for a rater to provide a rating—for example, even expert radiologists would give different opinions on the size and location of a lung nodule in an X-ray image depending on its specific features [36], even though biopsy can give an objective answer whether a suspicious region in the image is malignant or benign.

In tasks with cultural truth the usage of crowdsourcing is required in order to get a "universal" answer, representing the opinion of the general public. Asking just one rater will result in training data that represents only a biased and subjective opinion of a single individual. The usage of crowdsourcing is also vital in tasks with objective truth, for which ratings are difficult to provide as multiple raters can compensate for the mistakes of individuals. Also, getting the real answer in such tasks can be very expensive and invasive (for instance, conducting a biopsy on a tissue sample [136]), therefore, crowdsourcing is the only way to get the answer in a relatively fast and inexpensive way. Crowdsourcing is also widely used for the tasks with objective truth where the answer can be easily verified (such as textual entailment). In such tasks it is often possible to hire a single expert, who will perform the rating, however, crowdsourcing can make this process much cheaper and faster [8, 165, 168]. Some researchers leave out the cost component at all, concentrating only on the possibility of getting ratings fast enough [1].

Regardless of the task at hand, the process of crowdsourced corpus rating almost always looks the same. Typically, a single party interested in ratings posts all training instances online. The rating of each training instance is presented as a single task, which has to be solved by a single worker independently of other workers by submitting a rating (usually, by selecting from a set of predefined ratings). Each worker can perform as many tasks as he wants, usually he gets paid for every rating submitted, however, in some cases the rating can also take place voluntarily [22]. As a result, each training instance will have a number of ratings assigned to it by different raters. There usually is a certain budget, from which a fixed payment is paid for each rating collected. Involving multiple raters in rating each instance can make crowdsourced solutions quite expensive [87]. That is why the task of decreasing the overall cost of the rating collection receives a lot of attention

[55, 197, 205].

When all ratings are gathered, they are aggregated to provide *a prediction*, a single answer for every instance. Then these predictions are used as a training set (to train a classifier/predictor) or as a validation set (to measure the performance of a classifier/predictor that has already been trained). It is expected that the predictions are close to the *gold standard*, a set of true ratings for each instance that are not known in advance. A large volume of research reports that these predictions are indeed quite accurate [123, 127]. Applications where crowdsourcing was successfully used for rating tasks include computer vision [197], natural language processing [165] and medical imaging [136].

A typical scenario of collecting ratings is the following: every rater can rate a single instance once, and all raters do exactly the same task: provide a rating when being presented with an instance without interacting with other raters [55, 94, 136, 198]. A few researchers present complicated multi-stage rating processes: for instance, Dai et al. [42] proposed a framework where answers can be iteratively improved. They used recognition of handwriting as one of the motivating examples. In such a setup each instance (a handwritten sentence or paragraph) is presented to a rater who can leave some of the words unrecognised. Such partial recognition can be a great help to a second rater, who might be able to recognise the previously unrecognised words by context. One other interesting exception is the work by Fang et al. [60], who explored a model in which raters can teach each other.

Independent of the rating process details, there will always be noisy raters, who provide inaccurate ratings either because of a lack of expertise or in order to get payment without investing any effort. There are different quality control techniques that allow the detection of such inaccurate ratings and can eliminate them or compensate for them. These techniques can be divided into three groups, depending on the stage of the rating process at which they occur:

1. **Before the start**: before a rater can rate any instances, he has to go through a qualification test (for example, rating a few test instances for which the gold standard is already known [78, 172]). If a rater fails this task, he is not permitted to rate any

instances.

2. **After the finish (static estimation of rater reliability)**: any rater can rate as many instances as he likes. When all ratings are collected, a procedure is used to estimate rater reliabilities. When calculating predictions, ratings coming from the raters with high reliability have more weight than those coming from unreliable raters [136, 198].

3. **During the process (dynamic estimation of rater reliability)**: the reliability of raters is tracked dynamically as they rate instances. As soon as an unreliable rater is detected, he is not presented with new instances to rate [55, 197].

Currently there is no strong evidence in the literature that methods from the first group are actually beneficial. Heer and Bostock [78] report that qualification was able to reduce the proportion of invalid ratings from 10% to 0.4%, while Su et al. [172] were unable to find any correlation between rater performance in the qualification task and that of rating of actual training instances. At the same time, both static and dynamic techniques are widely used and have been proven to be successful [55, 136, 197]. We review state-of-the-art static and dynamic techniques in Sections 3.3.1 and 3.3.2.

3.3.1 Static estimation of rater reliability

The main idea behind static estimation of rater reliability is first to collect all ratings and then estimate the predictions. The simplest way of aggregating the ratings from raters is using some form of averaging (majority vote in classification and mean in regression), which was used in some early crowdsourcing work [165]. The problem with averaging is that all raters are considered to have the same degree of proficiency as their ratings contribute equally. It has been shown that approaches that take the differing skill of raters into consideration can lead to better results [136, 198]. While simple averaging is still widely used as a naïve baseline, a typical static approach calculates the prediction for an instance as the average of ratings coming from raters, weighted by their reliabilities.

Following the seminal work by Dawid and Skene [46], static approaches tend to use

an expectation-maximisation (EM) algorithm. Each iteration of this algorithm consists of two steps: estimating the rater reliabilities (E-step) and estimating the predictions (M-step). At each iteration the reliabilities are estimated more and more precisely, which also facilitates the more precise calculation of predictions. It is expected that the error of predictions will decrease as the algorithm moves from iteration to iteration. When a new iteration does not make any changes to reliabilities and predictions, the algorithm is considered to have converged and the predictions are reported as the result. According to Karger et al. [94], an EM algorithm is a heuristic without any rigorous guarantees about its correctness or its overall performance. They also state that it is impossible to predict in advance whether the EM algorithm will converge for a particular problem. In binary rating tasks EM algorithms can be avoided as many non-EM algorithms exist for such problems, for instance an alternative formulation of a binary support vector machine optimisation problem [48]. However, for more difficult settings such as multi-class classification or regression, very few non-EM algorithms have been proposed. That is why EM algorithms are widely used in such tasks, despite the shortcomings mentioned above. One notable exception is the work by Karger et al. [94] where a non-EM approach based on low-rank matrix approximation is used for multi-class classification. Nevertheless, Liu et al. [113] demonstrated that in some cases this approach performs even worse than a very basic majority voting baseline.

An EM algorithm typically does not require every rater to rate every instance, although the degree of rating sparsity required for the approach to work is usually not specified explicitly. Rodrigues et al. [140] briefly discuss possible problems that can arise from sparse ratings. They highlight that an EM algorithm can require fitting a large number of parameters, especially in multi-class classification settings. This means that a large number of ratings from every rater is required to estimate his reliability correctly. This suggests that EM algorithms can produce inaccurate results if ratings are sparse. The algorithm by Jung and Lease [90] deals with this problem by estimating sparse ratings and is able to perform successfully even when only 0.4% of the total number of ratings are provided. However, it is suited only for binary classification, and it is not clear if

a similar approach can be developed for more complicated scenarios such as multi-class classification or regression.

Many static algorithms assume that all instances have the same degree of rating difficulty. However, this does not always hold in real life: for example, an average radiologist might make wrong conclusions from a "difficult" medical image, while a skilled one will provide a correct answer. Also, a certain image can be "easier" for a particular radiologist as he might have good experience in that particular category of medical images [205]. Some approaches take these factors into account, resulting in models with a large number of parameters and, therefore, more prone to overfitting. However, such models usually are not much better than alternative algorithms that ignore the varying difficulty of instances. For instance, using a recent approach by Audhkhasi and Narayanan [11] which models the varying performance of raters across different instances results in only a minor improvement compared to the algorithm of Raykar et al. [136] where all instances are considered to be equally difficult.

Many static techniques require that every instance has a vector of features associated with it. In algorithms for estimating rater reliability features are often used in modelling the varying performance of raters described above. It is rarely stated, but these features are expected to be highly informative. For instance, Xiao et al. [203] assume that if two instances are close in the feature space, they should also have similar predictions. Xiang Liu and Memon [202] make an even stronger assumption by placing all instances in 1D or 2D feature space and assuming that instances belonging to different classes can be linearly separated. One of the shortcomings of this requirement is that we might not know enough about the data to be rated and thus would not be able to design a set of features.

Most research in the static estimation of rater reliability control covers binary classification techniques. Some researchers have also considered multi-class classification, including the work by Raykar et al. [136] who suggested an algorithm for tasks where classes are ordinal. At the same time, the problem of regression receives little attention. To the best of our knowledge, the only static technique that can work for regression tasks when no

Source	Binary classification	Multiclass classification	Regression	Instance difficulty	Features are required
[11]	+	+		+	+
[15]	+	+		+	
[43]	+				
[46]	+	+			
[48]	+				+
[75]			+		+
[90]	+				
[91]	+			+	+
[92]	+				+
[94]	+	+			
[95]	+			+	
[113]	+				
[112]	+	+			
[131]	+			+	+
[136]	+	+	+		
[134]	+	+			
[140]	+	+		+	+
[185]	+			+	+
[196]	+			+	+
[198]	+			+	
[202]	+				+
[203]			+	+	+
[213]	+			+	+
[215]	+	+		+	

Table 3.2: Survey of static techniques for estimating rater reliability. It is shown for which tasks each algorithm is suited, as well as whether the algorithm needs instance features in order to work. A plus sign in the column "Instance difficulty" means that the algorithm models the difficulty of instances in some way.

instance features are available, is the approach by Raykar et al. [136][9].

A survey of modern static techniques is given in Table 3.2. It lists all the approaches mentioned in this section, describing the tasks for which they are suited. It is also noted which approaches require a set of features and/or consider the varying difficulty of instances.

3.3.2 Dynamic estimation of rater reliability

One of the problems with static estimation of rater reliability is that ratings are gathered from raters with varying expertise, so that a part of the rating budget is paid for inaccurate ratings. One solution to this problem is to delay payments until the process is over and reliabilities are calculated, and then pay only the raters who are deemed to be reliable. However, this approach has a problem in practice: the total costs can be unpredictable as the number of reliable raters is not known in advance. Dynamic estimation of rater

[9]Potentially, there is an opportunity to use techniques from other fields to perform the task of static quality control. For instance, it is not unlikely that techniques that rely on identifying outliers such as RANSAC [63] could also identify outlying ratings.

reliability approaches the problem of estimating predictions from a different angle. Static techniques allow all raters to rate as much as they want, while dynamic algorithms track the rater reliability as they work. As soon as an unreliable rater is discovered, he is not asked to rate anymore. Thus, most instances should get ratings from reliable raters only. Then, rating costs can be controlled by capping the maximal number of ratings per instance. In such a dynamic scenario the reliability of a rater is usually determined by how well he agrees with other raters who rated the same instances.

Currently, many dynamic techniques exist (Table 3.3). Some of them are situated within active learning frameworks, where learning takes place at the same time as the collection of ratings [47, 55, 205]. In such a setup the goal is to train a classifier/predictor rating only the most informative part of the dataset. Although other dynamic techniques require getting ratings for all instances in the dataset [93, 129, 197].

All dynamic techniques can be divided into two groups by the assumptions on rater availability:

1. **Constant availability**: every rater is available at any time to rate any instance [47, 55, 205]. In such a scenario it is possible to issue requests to particular raters to rate certain instances. All such approaches tend to work in a very similar fashion: every step of the process consists of picking an instance, selecting a rater to rate it and collecting the rating [33, 47, 55, 129, 181, 195, 201, 205]. The reliabilities of raters are updated online based on the ratings they provide, and the process is over when a sufficient number of ratings is collected.

2. **Intermittent availability**: raters are not constantly engaged in the rating process, they can leave and re-enter arbitrarily [34, 197, 216]. In this scenario it is infeasible to ask a rater to rate a particular instance as a time delay between the request and getting the rating usually occurs. It is also possible that the rater will not provide the rating at all. Thus, usually instances are presented to raters once they make themselves available to rate.

   One of the approaches to deal with intermittent availability, the algorithm by [197],

maintains three groups of raters arranged by their reliability: (i) "experts", reliable raters who make few mistakes, (ii) "bots" who provide random ratings without investing any actual effort, (iii) all other raters. When an instance gets picked for rating, the process stops for time T to wait for an expert. If he becomes available in this time, his rating is accepted, if not, the rating from the first rater available after T has passed is taken, providing that this rater is not a bot. If the instance has a sufficient number of ratings, the process moves to the next instance, if not, time T again is spent to wait for another expert. Another approach to intermittent availability by Chien-Ju Ho [34] is specifically suited for the situation when there are several types of tasks, for instance, requiring different levels of ability or different areas of expertise. When a rater becomes available, he gets assigned to the task of the type in which he showed the highest accuracy in the past. While it is in general assumed in intermittent availability scenarios that any rater can potentially re-enter the rating process in the future, Zou and Parkes [216] considered a situation when every rater becomes available only once. When a rater becomes available, he can rate as many instances as needed until the dynamic algorithm decides to stop taking ratings from him. After this the rater never becomes available again. Another intermittent availability approach by Ho et al. [79] not only tracks rater reliability, but also keeps scores denoting how confident the algorithm is in a particular reliability value. Ratings are accepted only from those raters who are definitely accurate or definitely noisy. At the end of the rating process, when predictions are calculated, the noisy ratings do not have any effect on predictions, because they get small weights. The algorithm does require each rater to rate a number of instances for which the gold standard is already known.

Algorithms for the scenario of intermittent rater availability can be divided into two classes: *instance-driven* and *rater-driven*. In instance-driven approaches [197] the rating process consists of picking an instance and assigning incoming ratings to it. When a sufficient number of ratings has been gathered, the next instance is selected. In contrast, rater-driven approaches [34, 79, 216] perform selection of instances for

each incoming rater separately. When a rater becomes available, instance(s) to rate are specifically selected for him.

The constant availability scenario faces only the problem of accurate reliability estimation, while intermittent rater availability also has to take time into consideration dealing with problems such as "if a relatively unreliable rater is available right now, should we ask him to rate or is it better to wait for a more reliable rater"? So, it is not only a problem of getting the lowest possible error with the lowest possible cost, but also collecting ratings as quickly as possible. It is interesting to note that papers proposing such "time-aware" approaches usually do not consider time as a factor in their measures of performance [34, 79, 197, 216].

Tasks with constant availability are relatively rare, though still can be encountered. For instance, Brew et al. [23] recruited a crowd of volunteers to rate a dataset for sentiment analysis in news articles. They presented a batch of articles to all raters every morning. Ratings had to be submitted during the whole day, before the next batch is presented, so the approach was not very time-sensitive. Such a setup was possible because the rating process was integrated into the raters' everyday news reading routine.

Clearly, the scenario with intermittent rater availability is more common: raters usually work through the Internet, so direct control over them might not be feasible. They might be in different time zones, which can cause problems with availability. Also, if a crowdsourcing platform is used it might be impossible to contact a rater directly to ask him to rate something right now.

Some state-of-the-art dynamic techniques have a number of limitations. For instance, they often are suited only for very specific types of tasks such as binary rating [79, 201, 216] and can not be easily adapted for multi-class rating or regression. The IEThresh algorithm [55], where this adaptation is very straightforward, is an exception. Some algorithms [34, 181] assume that the quality of every rating can be estimated instantly and independently of other ratings. In supervised machine learning it almost always is equivalent to assuming that there is an oracle that can provide a correct rating. However, if such an oracle is available, there would be no need to collect ratings in the first place [79].

Source	Binary classification	Multiclass classification	Regression	Availability of raters	Features are required
[33]	+	+		Constant	
[34]	+	+	+	Intermittent	
[47]	+			Constant	+
[55][10]	+			Constant	+
[58]	+	+		Constant	
[79]	+			Intermittent	
[93]	+	+		Constant	+
[129][11]				Constant	
[181]	+	+	+	Constant	
[195]	+			Constant	
[197]	+	+	+	Intermittent	
[201]	+			Constant	+
[205]	+			Constant	+
[216]	+			Intermittent	

Table 3.3: Survey of dynamic techniques for estimating rater reliability. It is shown for which tasks each algorithm is suited, as well as whether the algorithm needs instance features in order to work. Every algorithm works either in the conditions of constant availability (all raters are available all the time and provide ratings immediately) or intermittent availability (raters can enter and leave the rating process at arbitrary times).

There also are other challenges, for instance, in the approach by Welinder and Perona [197], after providing a rating, a reliable rater can not continue the rating process until the next instance is selected for rating. The duration of this delay can be unpredictable and depends on the availability of raters as well as on the selection of T, the waiting time period. It is not clear how to overcome this in practice as the authors present results based not on an actual rating experiment, but on data collected in advance. Some research also assumes that a certain knowledge, such as the parameters of certain statistical distributions, is known before the rating process starts. For instance, Welinder and Perona [197] required knowing the distribution of rater errors. They assumed that it is Gaussian, but it is not clear how to select the σ parameter of this distribution and how accurate this estimate needs to be. Kamar et al. [93] described an approach that uses the distribution of errors too, but they propose to infer it from ratings collected earlier for the problem at hand from other raters. This however, suggests some other method of collecting ratings has already been used.

As can be seen, there is a big variety of quality control techniques, both static and dynamic. However, the literature lacks the guidelines where which technique—static or dynamic—is more preferable. In order to come up with some recommendations, a cat-

[10] Can be easily generalised to multiclass and regression tasks.

[11] Works only for the tasks where raters have to compare a pair of instances among themselves.

egorisation of crowdsourced rating tasks would be beneficial. However, as rating tasks are usually very similar, most of the dimensions described above are not applicable for categorising these tasks. In Section 3.4 we propose a novel categorisation of rating tasks that helps to contextualise static and dynamic quality control techniques.

3.4 Proposed categorisation of rating tasks

A lot of categorisations reviewed in Section 3.2 concentrated on a very broad range of crowdsourcing problems, however, not all of the dimensions proposed are relevant for rating tasks. In this subsection we first review different dimensions and select those, which can be used in the categorisation of rating tasks and then propose new dimensions based on the review of rating tasks in the previous section.

3.4.1 Participant-related dimensions

Many participant-related dimensions seem to be quite irrelevant for rating tasks: for instance, raters are never at the *same location* at the *same time*, and their *role* always is similar. As can be seen, there are a few ways of approaching the motivation for raters in state-of-the-art research (*motivation*, *explicit/implicit work* and *payment strategy* dimensions), however, rating tasks mostly rely on raters who are providing ratings explicitly and for a payment. The difference between motivation strategies might be interesting if the problem of attracting raters is considered, but in the context of rating tasks, the motivation of workers is not that important as long as raters rate in reasonable time with reasonable costs. Therefore, for the sake of simplicity, we propose to substitute motivation and payment-related dimensions with a single one, denoting *a cost of a single rating*. The cost is somewhat related to *requirements for the crowd* and *human skill, which is required for the task*—the more specific requirements we have, the higher the cost will probably be. At the same time, there can be exceptions for this rule—for instance, even highly-skilled professionals can volunteer. As rating tasks can require different and sometimes quite specific skills, the dimension of *requirements for the crowd* is relevant for rating tasks as well.

3.4.2 Task-related dimensions

A rating task usually involves a choice between few alternative ratings or providing a numerical rating, therefore, all rating tasks would have the same *degrees of freedom*, *nature* and *design/type*. At the same time, the dimension of *complexity* allows to draw a distinction between different rating tasks, however, the creative task class proposed by Schenk and Guittard [142] would not be applicable to rating tasks, although, two other classes they proposed (simple, such as reCAPTCHA, and complex, knowledge intensive ones, such as translation of phrases), will be used in our categorisation. Considering that keeping the total time of rating the corpus as short as possible is one of the critical success factors in crowdsourcing, the *participation time* dimension also is important for rating tasks.

3.4.3 Process-related dimensions

As we have pointed out in Section 3.3, the process of crowdsourced corpus rating almost always happens in the same way. There is no any *competition* or *interaction*, the *degree of collaboration* is zero, the *degree of automation* is very high, the *task/request cardinality* is the same, as well as the *management of the process* and a method of *aggregation of worker results* (several ratings are combined into a single one). The *principal beneficiary* can have a certain impact on the rating task, however, it is tightly related to the motivation, which we discussed above. The only process-related dimension, which is very relevant to rating tasks is *quality control*, which can be either static (estimation of rater reliability and target ratings at the end of the process) or dynamic (doing it when raters rate the training data). It is especially interesting considering the absence of recommendations on which approach to choose for which task.

3.4.4 Proposed dimensions

In addition to the dimensions of *cost*, *requirements for the crowd*, *participation time* and *quality control*, there are more dimensions that can draw a distinction between different rating tasks. For instance, in some tasks it is possible to verify the answer quite easily

(such as the tagging of an image very often can be checked by brief looking at the image), in some it requires a complicated procedure (cancer diagnostics), while in some it can not be done at all due to the absence of ground truth (rating of emotion or other tasks with cultural truth). If the answer can be verified easily, it allows a small subset to be created quickly, which can be used to measure the accuracy of raters by comparing their ratings to the ones from the subset. Therefore, the dimension of the *ease of the verifiability of the correct answer* is relevant for rating tasks. One more important dimension is the level of consensus, i.e. rater agreement on ratings. The consensus usually is quite high in some tasks, such as text entailment (Zeicher et al. [211] report the value of inter-rater $\kappa = 0.79$[12]) or recognition of speaker's accent ($\kappa = 0.78$ [159]). However, some other tasks, such as emotion recognition, can lead to much smaller values. We calculated the value of κ on 17 raters, who rated the VAM corpus [74] on evaluation dimension (how positive or negative the speaker is) and got a value of only 0.11. The degree of consensus for rating tasks is important, because it can make quality control easier or more complicated. If the consensus is high, it means that most raters would give the same answer and, by the general assumption of crowdsourcing, it will be close to the real answer. Therefore, raters, who disagree with the majority most times, are likely to be inaccurate ones and their ratings are likely to be noisy. The same reasoning would not be applicable in a task with low consensus, as it will be much more difficult to identify "the majority", which gives similar answers. Also, the differences between raters often can be explained by the nature of the task, not by differing accuracies of raters.

3.4.5 Categorisation

Concluding the discussion above, we propose to use the following dimensions to categorise crowdsourced rating tasks:

1. Cost of a single rating:

    - Free

[12]The value of κ-statistic can be in the range from 0 (no agreement at all) to 1 (perfect agreement all the time)

- Micro (a micropayment of a few cents)
- Payment—more than a micropayment

2. Size of the potential crowd (adapted approach by Corney et al. [38]):
   - Huge—the task can be completed by anybody
   - Big—certain not unique skill is required
   - Small—a certain expert specialisation is required
3. Complexity of the task:
   - Low—the rating is almost a passive process, no significant attention is required
   - Medium—the rating involves some amount of attention and thinking
   - High—high demands in mental and intellectual sense
4. Time, required to provide a rating (participation time):
   - Short—a few seconds are required
   - Long—rating might take 10 minutes or more
   - Medium—everything in between
5. Verifiability of the correct answer:
   - Impossible—tasks with cultural ground truth
   - Easy—the answer can be verified easily (picture tagging)
   - Difficult—the verification of the answer is possible, but requires a time and/or cost consuming procedure (such as biopsy in cancer diagnostics)
6. Degree of consensus:
   - Low—raters would tend to give different answers
   - High—raters would tend to agree on answers
7. Quality control policy:

- Static—the estimation of rater reliability and target ratings takes place only when all ratings are collected
- Dynamic—the estimation of rater reliability and target ratings takes place dynamically, as raters rate training instances

Dimensions *Verifiability of the correct answer* and *Degree of consensus* are introduced in this thesis for the first time, to our best knowledge. All other dimensions are adaptations of dimensions already mentioned by other researchers.

Some possible rating tasks and the according values of dimensions are given in Table 3.4.

At the first glance it might seem that some of those dimensions are correlated. For instance, the more specific requirements for the crowd we have, i.e., the more unique raters are, the higher is a cost of a single rating. It might be true in many cases, however, if we look at the task of translation from one relatively common language to another (both belonging to *Broad* category on *Requirements for the crowd* dimension), the cost might be different depending on the pair of languages at hand. Heer and Bostock [78] propose to tie the cost of the rating to the minimal wage of the country, where the rater is located. As native speakers of some languages tend to live in one particular country, all of them will expect a certain price per rating depending on the minimal wage, which can vary quite significantly from country to country. However, if raters are experts from a narrow area, the link between the participation time and cost of a single rating would not be as straightforward—an expert radiologist can charge for a minute of his time much more, than a person who is solving CAPTCHAs. The correlation between cost and complexity of the task also would not hold true in all cases—a relatively non-demanding task of filling in the demographic information cost 0.10 USD in the work by Mason and Watts [118], but Ambati et al. [8] paid 0.015 USD for a translation of a sentence, which is more complex compared to filling in a form. Requirements for the crowd and complexity of the task also would not be always correlated—the complexity of a translation from one language to another is the same, however, the size of potential rater crowd can be quite different. The complexity of the task and participation time also would not necessarily be correlated,

Table 3.4: Examples of crowdsourced rating tasks, categorised along the dimensions proposed

Task	Cost of a single rating	Size of the potential crowd	Complexity of the task	Time, required to provide a rating	Verifiability of the correct answer	Degree of consensus	Quality control policy
Rating of products on Amazon.com or similar services	Free	Huge	Low	Short	Impossible	Low	Dynamic
reCAPTCHA	Free	Huge	Low	Short	Easy	High	Static
Sentiment analysis in blog posts	Micro/Payment	Big	Medium	Medium	Impossible	Low	Dynamic
Translation of short phrases (common languages)	Micro	Big	Medium	Medium	Easy	Low	Static
Translation of short phrases (rare languages)	Payment	Small	Medium	Medium	Easy	Low	Dynamic
Image tagging (identifying the season or object depicted)	Micro	Huge	Low	Short	Easy	High	Static
Textual entailment	Micro	Huge	Low	Short	Easy	High	Static
Medical imaging	Payment	Small	High	Long	Difficult	Low	Dynamic
Emotion rating	Payment	Big	Medium	Medium	Impossible	Low	Dynamic

because there might be tasks, which take the same time, but have a different cognitive demands—for instance, transcription of an audio record and diagnosing a patient based on an X-ray image. The verifiability of the correct answer and the degree of consensus also might seem to be related—the more difficult the verification is, the more likely raters are to give different answers, however, it is not true in all tasks. If we take gender recognition from a voice recording as an example, no information about the speaker is available (so the ground truth can not be verified), however, raters show very high degree of agreement on this task [159].

Values on all of those scales can be determined by the nature of the task, with the only exception of *Quality control policy*. Some tasks require using a dynamic approach, for instance, estimating a true rating of a product based on ratings submitted by customers. As new customers are buying product, they leave ratings on the seller's web-site (such as Amazon), and the process of the rating collection is ongoing, i.e. there is no point when the collection can be considered finished. For the majority of tasks both the static and dynamic approach can be applied, with the static approach potentially leading to better accuracy of target ratings, as it operates on more information compared to any dynamic approach. However, in some cases a static calculation of ratings might be significantly more expensive and take longer to collect ratings than a dynamic one, so the latter should be preferred. We recommend using a dynamic approach in the following cases:

1. When the cost of a single rating is bigger than a micropayment. The higher the cost is, the more benefit will be in using a dynamic approach, as it would not require paying for all ratings.

2. When the participation time is long or medium, the longer it is, the more beneficial a dynamic approach will be. The dynamic approach will be based on a smaller number of ratings, therefore, the collection process will finish faster. However, if many raters are potentially available, the time gain compared to static approach might not be too significant.

3.5 Multi-armed bandits

The dynamic estimation of rater reliability, to which this thesis is devoted, can be represented as a *multi-armed bandit* task, a mathematical abstraction representing the task at hand as a gambling machine. In this section we describe multi-armed bandits in detail, as well as review how they have been used to date for solving different crowdsourcing tasks, including the estimation of rater reliability.

A multi-armed bandit, formulated for the first time by Robbins [139], is an abstraction of a problem of choosing the best option from a certain set of options. Such tasks arise in many application areas. For instance, finding the best medical treatment from a number of options, where the best treatment is the one which causes the least discomfort for the patient [77], although the amount of discomfort that will be caused by a treatment cannot be known in advance. Another problem, where multi-armed bandits have been applied, is choosing the fastest network information provider out of a set of providers [187], where the performance of providers cannot be known in advance of utilising them. A multi-armed bandit represents such problems as a gambling machine with multiple arms, each of them corresponding to a single option (a drug or a service provider in the examples above).

In this section we describe different ways the multi-armed bandit task is formulated in state-of-the-art research (Section 3.5.1). We then continue with the description of algorithms that are used to find the best arm (Section 3.5.2).

3.5.1 Formalisation of the task

Following the formulation given by Vermorel and Mohri [187], a multi-armed bandit represents a problem as a k-armed slot machine. At each of T trials one arm is pulled and a numerical reward is received—the higher the reward, the better the arm. The task is to maximise the sum of rewards, collected during the T trials. This can be achieved by finding the best arm in as few pulls as possible and then continuously pulling it, until all trials are over. In general, there are no limitations on a number of pulls for each arm. The rewards received by pulling each arm at each step of the algorithm are recorded and used to estimate the quality of each arm. Bubeck and Cesa-Bianchi [24] mention three main

formalisations of the bandit problem depending on the nature of rewards:

1. **Stochastic bandits:** it is assumed that the reward associated with pulling an arm is a variable drawn from a fixed, but unknown, probability distribution. Stochastic bandits were covered in state-of-the-art research to a great extent, and are much more widely used than other two formulations. The overwhelming majority of research using multi-armed bandits for crowdsourcing also uses stochastic formulation.

2. **Adversarial bandits:** the rewards associated with each arm are set by a special mechanism called *adversary* or *opponent*. He can turn an arm that was very good in the past into an inferior one, which makes the task of finding the best arm substantially more difficult than in the stochastic case. Research in adversarial bandits tends to be more focused on theory than on immediate practical applications [24].

3. **Markovian bandits:** in this formalisation each arm is associated with a Markov process, each with its own state space. When an arm is pulled, the distribution from which the reward is drawn depends on the current state of the Markov process [24]. When an arm is pulled, it's state changes, while the state of other arms remains unchanged. According to Bubeck and Cesa-Bianchi [24], Markovian bandits are a standard model in the areas of economics and operation research. However, they tend not be used in computer science community.

There are also many different special cases of multi-armed bandit problems. They usually consider some specific condition that is a feature of certain tasks. Most of them can be used in all three formulations mentioned above, however, some of these bandits are only suited for a particular formulation of a bandit problem. Here are some of the examples mentioned by Bubeck and Cesa-Bianchi [24]:

- **Contextual bandits:** each arm's reward depends on a certain side information or context. A personalised news recommendation engine can be mentioned as an example of such a task. There always is a pool of articles, and some of them might be displayed to the user in a shortened format, for instance, including only a headline,

an illustration and a few introductory sentences. Displaying an article corresponds to pulling an arm; if the user clicks on it to read more, the reward is equal to one, zero otherwise. It is clear that different users will be interested in different news stories. Side information about both users and articles can be used to improve the performance of the recommender. For instance, if the user in the past read a lot about pop-music, he might be more inclined to click on such news also in the future.

- **Dueling bandits:** at each trial two arms are always pulled instead of one. The user can not observe the exact values of rewards for both arms: only the relative performance is known, for instance, which arm had a higher reward.
- **Many-armed bandits:** these are stochastic bandits that have an infinite or very large number of arms.
- **Restless bandits:** Markovian bandits, where states of all arms change after an arm is pulled.
- **Sleeping bandits:** the set of available arms changes from trial to trial, some arms can "fall asleep" from time to time.

Different formalisations and types of tasks use different algorithms to find the best arm. Section 3.5.2 describes such algorithms in a greater detail.

3.5.2 Algorithms

Multi-armed bandit algorithms differ in the way the quality of an arm is calculated and used. Caelen and Bontempi [29] propose a taxonomy to group the large variety of multi-armed bandit algorithms in the literature into four categories:

1. The Gittins index policy proposed by Gittins [67] is one of the oldest multi-armed bandit strategies and considers the problem of determining the optimal arm as a dynamic programming problem. It is rarely used in current research due to a number of computational difficulties and logical inconsistencies [156].

2. Semi-uniform strategies focus on separating the *exploration* and *exploitation* aspects of multi-armed bandit approaches. Vermorel and Mohri [187] present a number

of semi-uniform algorithms, and their work with both synthetic and real data suggests that the ϵ-first strategy is the best among semi-uniform approaches. The ϵ-first strategy performs exploration (pulling a random arm) at each of the first $\epsilon \cdot T$ trials and performs exploitation (pulling the arm with the highest mean reward up to date) the rest of the time. The parameter ϵ denotes the proportion of trials for which exploration should be performed. The performance of the very straightforward semi-uniform strategies is sometimes difficult to beat even with other more sophisticated multi-armed bandit algorithms [187].

3. UCB strategies calculate an upper-confidence bound (UCB) for the mean reward of each arm and pull the arm with the highest UCB. In semi-uniform techniques each trial is either exploitative or explorative, while such distinction does not exist in UCB techniques. By introducing a confidence bound, UCB strategies prevent situations where non-optimal arms are continually favoured because they have received a small number of high rewards and where arms that have not been explored (pulled) enough early in the process continue to be ignored. Numerous algorithms have appeared in the last years including UCB-V [13], MOSS [12] and DMED [80]. Recent work [65] has proposed the KL-UCB algorithm (KL stands for Kullback-Leibler divergence used in the algorithm) and shown that it outperforms many of the existing UCB approaches on a range of artificial and real-life tasks. The KL-UCB algorithm starts with pulling all arms once in order to get an estimate of arm quality. Once every arm has been pulled exactly once, the algorithm then pulls the arm with the highest reliability to date at all following trials. The reliability r_a of arm $a \in A$, where A is the set of all arms, is calculated in the following way:

$$r_a = \max \left\{ q \in [0,1] : d\left(m\left(a\right), q\right) \leq \frac{\log\left(\log\left(t\right)\right)}{n} \right\}, \tag{3.1}$$

where n is a number of times the arm a was pulled to date, $m(a)$ is the mean reward[13] obtained by pulling the arm a (rewards are in the $[0,1]$ interval), t is the number of a

[13]The way how reward is calculated depends on the task at hand. Formula 5.1 shows how it is calculated for the task of dynamic estimation of rater reliability.

current trial, and d is the Bernoulli Kullback-Leibler divergence given by

$$d(p,q) = p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}, \tag{3.2}$$

with, by convention, $0 \log 0 = 0$ and $x \log \frac{x}{0} = +\infty$ for $x > 0$.

Kullback-Leibler divergence is usually used to measure similarity between two random number distributions, but in KL-UCB algorithm it measures the similarity between two numbers, p and q. Number p is an estimate of the reliability of a particular rater, calculated as his mean reward to date. The formula 3.1 looks for q that is the UCB for a given p.

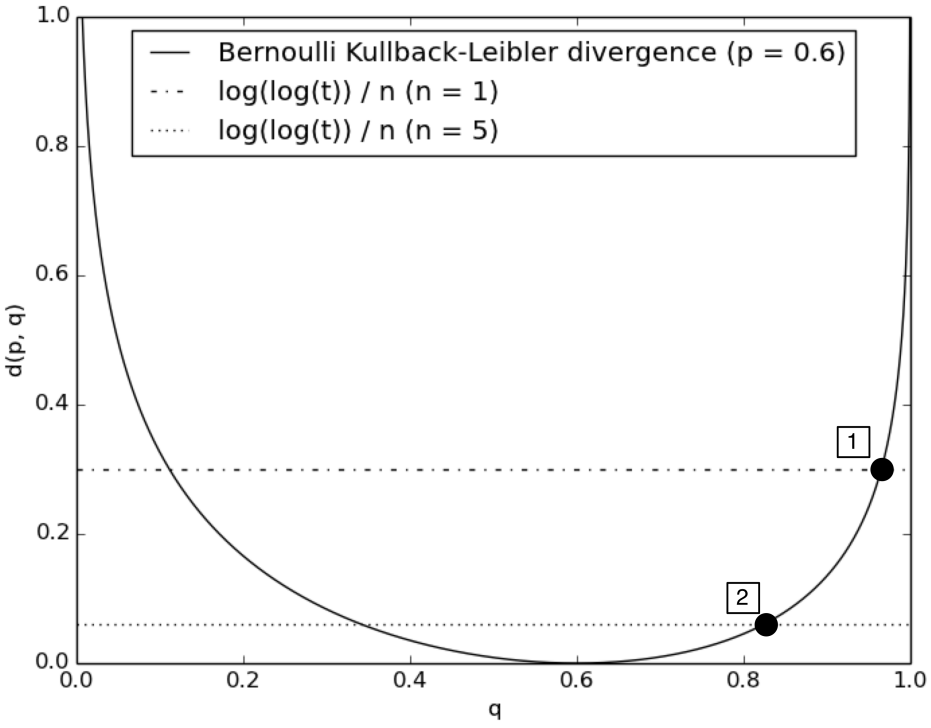


Figure 3.2: Illustration of how KL-UCB algorithm estimates the UCB for $m(a) = 0.6$.
.

Figure 3.2 provides an example of KL-UCB algorithm work when $m(a) = 0.6$. The Kullback-Leibler divergence (the left side of the inequality in formula 3.1) is plotted as a solid line. Let us assume that the estimate of $m(a)$ is based on one observation, i.e. the rater has received one reward to date ($n = 1$). The right side of the inequality in formula 3.1 is represented as a straight dash-dotted line. The output of the formula 3.1 is the X-coordinate of the right-most point where the Kullback-Leibler divergence curve

crosses the line representing the right side of the inequality (Point 1), thus, the UCB for the estimate of $m(a) = 0.6$ is approximately 0.96. As the estimate is based on only one reward, the UCB is quite far from the estimate. However, in the situation where the same estimate is based on five points, the UCB will be much closer to the estimate. The lines then will cross at Point 2, which means that the UCB will be equal to 0.82.

4. Probability matching strategies estimate the probabilities of each arm being the optimal arm and pull the arm with the highest probability. For instance, the SoftMax strategy [114] uses an algorithm that is similar to simulated annealing. These approaches tend to be less effective than the other categories [187].

Algorithms from each group can usually be applied to any formalisation of the bandit task. However, according to Bubeck and Cesa-Bianchi [24], Gittins indices are more suited for Markovian bandits, while one of the probability matching strategies, Exp3 [14], is specifically designed for adversarial conditions. For stochastic bandits there are algorithms available from all categories, but semi-uniform strategies and UCB strategies usually tend to lead to better results than algorithms from other groups [187].

The research into using multi-armed bandits for crowdsourcing is almost entirely concentrated on the stochastic setting, which makes it possible to use all variety of algorithms mentioned above. In Section 3.6 we review main works where multi-armed bandits were applied to some tasks in crowdsourcing.

3.6 Multi-armed bandits in crowdsourcing

Multi-armed bandits have been used for tasks related to crowdsourcing quite widely. For example, Singla and Krause [161] used a semi-uniform and a UCB-based strategy to choose an optimal pricing policy. The work by Chen et al. [33] uses multi-armed bandits to select the order in which instances should be presented to the raters. They used a Gittins index policy, but also acknowledged that it is computationally expensive. Tran-Thanh et al. [181] extended the ϵ-first algorithm to estimate the quality of workers. The task they considered was picking workers who can create computer programs of a high quality. In their setup,

every worker can perform only a limited number of tasks that is known in advance. As in the original ϵ-first, they first do exploration by spending a portion of the budget by asking all workers repeatedly without calculating or considering their quality. When exploration is over, the average rewards are calculated for each rater. Then the number of tasks to be assigned to each worker is calculated by representing the task assignment as a *bounded knapsack* problem. The exploitation phase consists of making this assignment and waiting for the results. A feature of their approach is the assumption that the quality of every answer can be evaluated and validated independently by an expert, which can be feasible in some domains such as estimation of code quality.

At the same time, in many areas, including gathering ratings for supervised machine learning, the oracle that can validate the quality of single ratings is not available. Otherwise, it would have been possible to obtain ratings from that oracle without asking actual human raters. The IEThresh algorithm [55] is specifically suited for such tasks. The IEThresh is originally formulated for binary tasks, and it asks several raters to rate an instance. Then the majority vote is taken as a prediction. If a particular rater's rating is equal to the prediction, he gets a reward of one, zero otherwise[14]. The reliability r_a of a rater a is calculated as

$$r_a = m(a) + t_{\frac{\alpha}{2}}^{(n-1)} \frac{s(a)}{\sqrt{n}}, \tag{3.3}$$

where n is a number of rewards received by rater a to date, $m(a)$ and $s(a)$ are respectively a mean and a standard deviation of rewards of rater a to date and $t_{\frac{\alpha}{2}}^{(n-1)}$ is a critical value for Student's t-distribution with $n-1$ degrees of freedom at $\frac{\alpha}{2}$ confidence level. In all our experiments $\alpha = 0.05$.

When an instance requires to be rated, several raters are chosen, based on their reliabilities. The choice of raters depends on how close their reliabilities are to that one of the top rater. At each round all raters who have reliability at least $\epsilon \cdot \max r_a$ are invited to rate ($0 > \epsilon > 1$). The smaller the value of the ϵ parameter, the more raters are selected at every round on average. It should be pointed out that the meaning of the ϵ parameter in

[14]In order to use this algorithm for rating tasks, in the experiments in this thesis the reward was calculated as in Formula 5.1.

IEThresh is different than that in ϵ-first. In IEThresh it influences the threshold for rater reliability, but in ϵ-first it determines the length of the exploration phase.

IEThresh has never been explicitly formulated as a multi-armed bandit algorithm, but it has all features of a UCB-based strategy: each rater represents an arm, and reliability for each arm is calculated as an upper-confidence bound for mean reward to date.

As can be seen, there are a variety of multi-armed bandit algorithms, and many of them have been used for different aspects of crowdsourcing tasks. However, the task of estimating rater reliability using multi-armed bandits when the accuracy of each rating cannot be independently verified received relatively small attention, especially, for the tasks where ratings are not binary values.

3.7 Conclusions

This chapter was devoted to crowdsourcing. Namely, many approaches to defining crowdsourcing were discussed, also distinctions between crowdsourcing and other related areas (such as human computing) were drawn. We reviewed multiple ways to categorise crowdsourcing tasks, however, many of them were inapplicable to corpus rating tasks, which are the subject of this thesis. Hence, a new categorisation specifically suited for such tasks was suggested and illustrated with some examples. This categorisation allows recommendations to be made about the quality control policy to be used in the task. More precisely, dynamic techniques (i.e. when rater reliability is estimated as raters rate, compared to static approach when it is done at the end of the rating process) can be recommended for the tasks where the price for a single rating is high, as well as for the tasks where participation time is long.

We also reviewed the main ways to formulate a multi-armed bandit task and main algorithms to solve it, i.e. to find the best arm as quickly as possible. The dynamic estimation of rater reliabilities can also be considered as an multi-armed bandit task where the goal is to find the best raters and to solicit ratings from them for all instances in a dataset—or to maximise the reward by pulling the best subset of arms until all trials are

over.

The state-of-the-art research often uses multi-armed bandits for crowdsourcing, however, the task of estimating rater reliability in such a setup when ratings are not necessarily binary values has not received a lot of attention. In the next section we introduce the main aspects of the methodology that later will be used to propose and evaluate an approach based on multi-armed bandits that is suited for binary, multi-class and ordinal classification, as well as regression.

Chapter 4

Experiment Methodology

The goal of this thesis is to develop a dynamic approach to the estimation of rater reliability that can be used in real-life tasks. The detailed results of the experiments performed are reported in Chapters 5–8. This chapter introduces the methodology used in the experiments. Four experiments were conducted:

1. **Dynamic estimation of rater reliability in the scenario of constant rater availability** (Chapter 5) with the main goal being evaluating whether multi-armed bandits can track rater availability better than a state-of-the-art baseline IEThresh, and a naïve approach where random raters were asked to rate every instance. In this experiment we used a scenario, where every rater was available immediately after being asked to provide a rating.

2. **Bootstrap issue in the scenario of constant rater availability** (Chapter 6) where we paid particular attention to the initial stage of the rating process, where bootstrapping takes place. At this stage multi-armed bandits mostly work in exploration mode, trying to learn rater reliabilities precisely. As a result, many noisy raters are asked at this stage, only in order to learn that they are not reliable. We investigated whether it is possible to detect these exploration-phase instances and gather additional ratings for them later in the process, in order to improve the quality of predictions.

3. **Dynamic estimation of rater reliability in the scenario of intermittent**

rater availability (Chapter 7), in which we evaluated whether multi-armed bandits can be used if raters can become available only from time to time. For this task we developed and evaluated the DER^3 (Dynamic Estimation of Rater Reliability in Regression) approach.

4. **Real-life evaluation of dynamic estimation of rater reliability (Chapter 8)**, where we used the DER^3 approach on Amazon Mechanical Turk.

This chapter is organised as follows. Datasets used in the experiments are covered in Section 4.1. Section 4.2 describes the approach to measuring performance, while Section 4.3 concludes the chapter.

4.1 Datasets

Most of the experiments conducted in this thesis simulate the rating process instead of using real raters on a crowdsourcing platform. An approach to estimating rater reliability dynamically requests ratings one by one, potentially, from any rater for any instance; these ratings are taken from a dataset of pre-rated instances. In order to make such simulation possible, datasets in which every instance is rated by all raters is required, so that every rating that could possibly be requested is available within the simulation. In many cases, especially, if the ratings were collected using Amazon Mechanical Turk or similar platform, the dataset is sparse: every instance has only a small number of ratings associated with it, also, every rater rates a relatively small number of instances. In order to use such a dataset in our experiments, we had to extract a portion of it, where every instance was rated by anyone. The following datasets were used:

1. **Vera am Mittag German Audio-Visual Emotional Speech Database**[1] (VAM), which contains non-acted video recordings of a talk show, divided into short segments collected by Grimm et al. [74]. Each speech segment is rated on three continuous dimensions. Ratings on all dimensions are in the set $[-1, -0.5, 0, 0.5, 1]$. The three dimensions are *activation* (how active or passive the recording is), *evaluation* (how

[1] http://emotion-research.net/download/vam

positive or negative it is) and *power* (how dominant the speaker is). A part of this corpus is rated by only six raters; however, 478 speech instances have been rated by 17 raters. The bigger number of raters represents crowdsourcing conditions more precisely, so we used the 478 instances as a dataset in our experiments. Following Schuller et al. [153], we extracted 384 acoustic features from the recorded speech instances, which included different functionals of pitch and energy, harmonics-to-noise ratios and Mel-frequency cepstrum coefficients. A separate dataset was created for each dimension (named *VAM_Evaluation*, *VAM_Activation* and *VAM_Power*).

Our initial experiments revealed that the performance of all 17 raters, who rated the VAM corpus, was very similar. To investigate this, we compared the predictions calculated by applying the algorithm by Raykar et al. [136] (which takes into consideration the varying rater reliability) to the predictions calculated by computing the mean of the ratings provided by all raters (which assumes that all raters have the same reliability). If the reliability of raters is varying, these two approaches should give significantly different results [198]. However, the average absolute difference between predictions in these two sets for the VAM datasets was 0.03 (for *VAM_Activation*) and 0.02 (for *VAM_Power* and *VAM_Evaluation*), which is 1.5% and 1% respectively of the whole $[-1, 1]$ scale. The absence of variance in ratings was also confirmed by analysis of the rater reliability measures produced as a by-product of the rating aggregation approach by Raykar et al. [136], which showed that almost all raters were equally reliable. This means that any approach to rater selection would produce similar results, which was also supported by our initial investigations. To introduce some variability into the ratings, so as to distinguish between the performance of the different rater selection approaches, we added 10 additional noisy raters to the dataset. The ratings for these noisy raters were generated by adding a random noise term, from a Gaussian distribution, to the actual rating of each instance[2]. This was similar to the approach adopted by Raykar et al. [136].

[2]Variance of the noise term was equal to 1 for *VAM_Activation* and *VAM_Power* and to 0.65 for *VAM_Evaluation*.

2. **BoredomVideos**: A dataset based on the work by Soleymani and Larson [166] who pursued a task of annotating videos to train algorithms for predicting viewer boredom. Raters had to evaluate how boring a particular video is on the [1, 9] scale. We extracted a small dataset of 45 videos annotated by the same 10 raters from the original dataset.

3. **ImageWordSimilarity**: A part of a corpus for the task of evaluating the semantic similarity between a word and an image [109]. Such data can be used in automatic image annotation, image retrieval and classification, as well as in other areas. We used the ratings submitted by 10 raters for 83 image-word pairs. All ratings were on $[0, 10]$ scale.

4. **Jester dataset**[3]: A dataset containing 4.1 million continuous ratings (on the scale [-10, 10]) of 100 jokes rated by 73,421 people [68]. Each joke is rated by a varying number of raters. A subset of ratings from 20 raters who have rated all 100 jokes was used as the experimental dataset.

5. **MovieLens dataset**[4]: A dataset consisting of 10 million ratings across 10,000 movies by 72,000 users. All ratings are in the [1, 5] range. We extracted a subset of 288 movies, each rated by the same 20 raters for our experiments.

Table 4.1 provides the overview of all datasets. As no true target ratings were available for any of the datasets, we calculated gold standard ratings for all instances using the rating aggregation approach by Raykar et al. [136] based on all of the ratings in each dataset. We refer to these as the *gold standard* ratings. The gold standard was not involved at any stage of the simulated rating process, rather, they were used solely for the purpose of the evaluation of different rater selection approaches.

As is the case with a lot of real-world data, all of the datasets described above were imbalanced. Figure 4.1 shows the distribution of the gold standard in each of the datasets used. Figure 4.1e show the distribution for *VAM_Activation*. There are very few recordings

[3]http://goldberg.berkeley.edu/jester-data/
[4]http://www.grouplens.org/node/73

Table 4.1: Datasets used in experiments with simulated rating process.

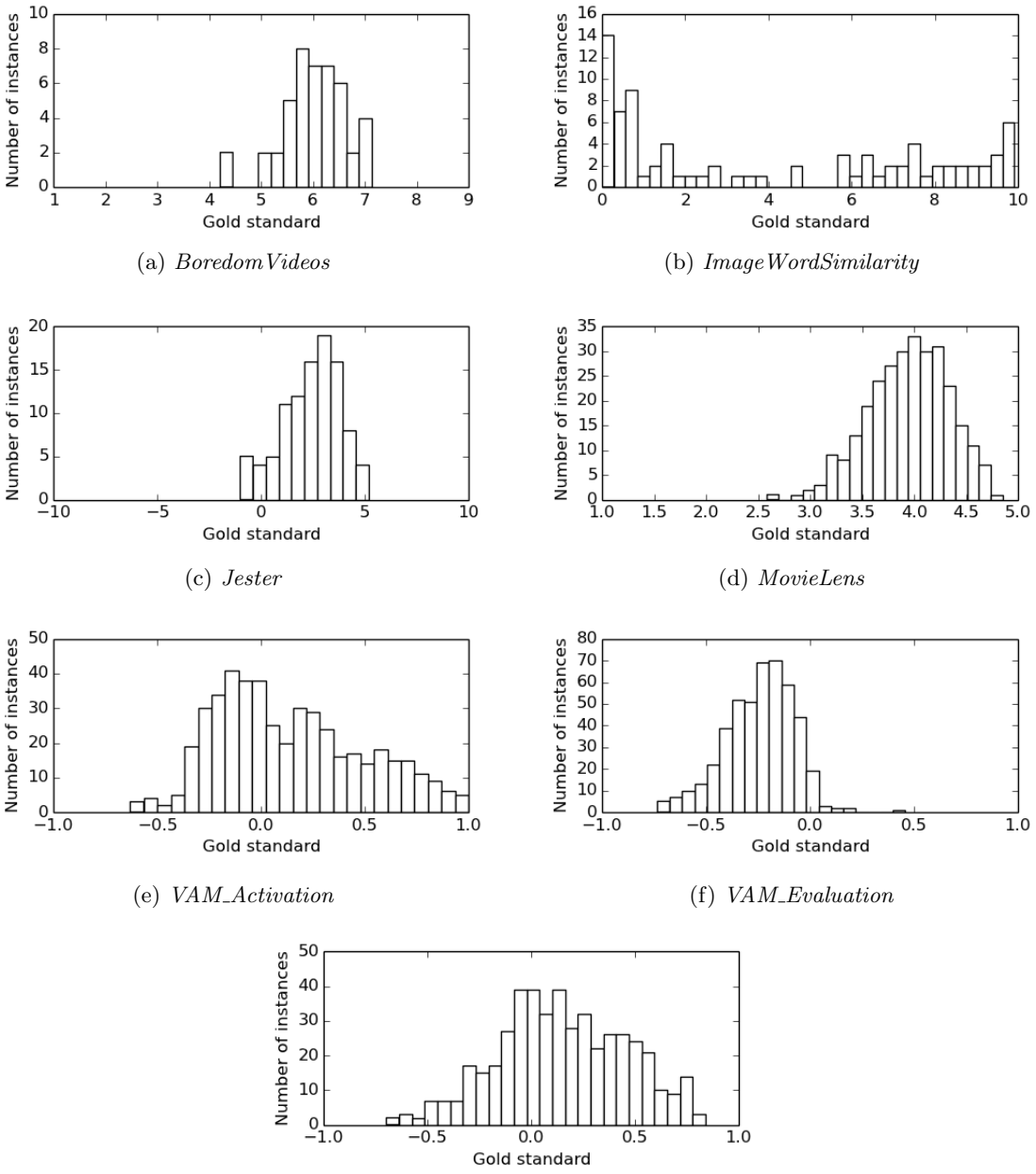
Dataset	Scale	Instances	Raters
BoredomVideos	Discrete, 10 points	45	10
ImageWordSimilarity	Discrete, 11 points	83	10
Jester	Continuous, [-10, 10] scale	100	20
MovieLens	Discrete, 5 points	288	20
VAM_Activation	Discrete, 5 points	478	17
VAM_Evaluation	Discrete, 5 points	478	17
VAM_Power	Discrete, 5 points	478	17

at the ends of the distribution depicting very active or very passive speech. In the field of emotion recognition from natural speech this situation is common and is encountered because most of natural speech tends to be neutral and non-emotional [103, 122, 170]. The situation is similar for all VAM datasets: most recordings are close to being neutral, but slightly skewed towards more active (Figure 4.1e), more negative (Figure 4.1f) and more powerful speech (Figure 4.1g). Grimm et al. [74] argue that such distributions are mainly due to the topics discussed in the talk show from the recordings of which the clips were extracted.

The videos in the boredom dataset seem to have a similar distribution. None of them were very boring or very exciting: the gold standard is centred approximately in the middle of the rating scale (Figure 4.1a). As can be seen from the gold standard ratings, both *MovieLens* (Figure 4.1d) and *Jester* (Figure 4.1c) contained movies and jokes that received relatively high ratings. *ImageWordSimilarity* is also imbalanced showing a lot of word-image pairs completely unrelated to each other (Figure 4.1b). However this dataset has the best coverage across all levels in the rating scale, a very different distribution to that seen in the other datasets. All experiments described in Chapters 5–8 use these datasets.

4.2 Performance measures

In our experiments we used three different performance metrics:



(a) *BoredomVideos*

(b) *ImageWordSimilarity*

(c) *Jester*

(d) *MovieLens*

(e) *VAM_Activation*

(f) *VAM_Evaluation*

(g) *VAM_Power*

Figure 4.1: Distribution of gold standard ratings in the datasets used in the experiments in this thesis.

1. **Cost**, which was measured as a total number of collected ratings.

2. **Time** between receiving the first and the last rating in the rating process (was used only for intermittent rater reliability conditions).

3. **Error** between predictions and gold standard ratings at the end of the process.

Our main measure is the average absolute error, expressed as a percentage of the original rating scale (for instance, an error of 0.15 on the *VAM_Activation* dataset would be equal to $0.15/2 = 0.075 = 7.5\%$). In order to gather more insight into how exactly errors are distributed, we often additionally considered the difference between predictions and gold standards in terms of a classification problem. The rating was considered as an ordinal classification problem, with classes being discrete values of the rating scale. The rating predictions were rounded to the nearest point on the rating scale, for instance, in VAM datasets 0.48 would be rounded to 0.5. The same was done with gold standard ratings. Then the average class-wise accuracy was used as a performance measure.

Ideally, an algorithm for the estimation of rater reliability should keep all three—cost, time and error—at a minimum. Although it is possible to analyse all these metrics separately, sometimes it is more convenient to aggregate them into a single numerical value. In order to do so, we used a multiplicative analytic hierarchy process (MAHP) which is recommended in the work by Triantaphyllou and Baig [182].

MAHP works in the following way.

Let us assume that there are m different approaches that have to be compared, each having associated values of cost (C_i), time (T_i) and error (E_i). First, all values are normalised in such a way that:

$$\sum_{i=1}^{m} C_i = \sum_{i=1}^{m} T_i = \sum_{i=1}^{m} E_i = 1. \tag{4.1}$$

Then for each approach a *preference score* is calculated by multiplying weighted normalised values of cost, time and error in the following way:

$$P_i = C_i^{w_C} \cdot T_i^{w_T} \cdot E_i^{w_E}, \tag{4.2}$$

where w_C, w_T and w_E are weights associated with the different criteria such that $w_C + w_T + w_E = 1$. The choice of particular values of weights depends on the task at hand, and in the experiments in this thesis all weights are equal to model a situation when all criteria are equally important. The lower the value of the preference score, the better the approach is as we aim for the lowest cost, time and error.

When the preference scores have been calculated, the approaches can be ranked and the average rank of each approach across all experiments can be determined. In order to check whether there are statistically significant differences between the ranks of the approaches, we use a well-known and established two-step procedure [49]. First, we use a Friedman test to check if any significant difference is evident between any of the approaches. Second, we apply a Bergmann-Hommel post-hoc test [64] to determine where exactly this difference lies. A typical result of such a post-hoc test is grouping approaches by their ranks, for instance, the group of the best approaches, the group of the worst approaches and the group of approaches in the middle. The differences between approaches belonging to the same group are not statistically significant even if their ranks are not exactly the same.

4.3 Conclusions

This chapter has covered the main parts of the methodology for the experiments described in the remaining chapters of the thesis. We described the datasets used and main performance measures: cost, time, and error (measured as average absolute error and average class-wise classification accuracy). Chapter 5 will describe the first experiment presented in this thesis, in which we look into the problem of estimating rater reliability dynamically in a scenario of constant rater availability.

Chapter 5

Dynamic Estimation of Rater Reliability in the Scenario of Constant Rater Availability

This chapter investigates whether multi-armed bandit techniques can be used to measure rater reliability dynamically, during the rating process. In this chapter, we cover the scenario of constant rater availability, i.e. every rater is available to rate immediately, at all times.

The remainder of this chapter is structured as follows. Section 5.1 describes the multi-armed bandit approach for the scenario of constant rater availability. Section 5.2 describes how this approach was evaluated. Section 5.3 presents the results of multi-armed bandit approach evaluation, and Section 5.4 concludes the chapter.

5.1 Approach

The goal of the rating process was to gather N ratings for each instance in a dataset from a set of potential raters, some of whom were believed to be better at the task than others. All training instances were rated one by one or in batches, using the most reliable raters to date. The approach to rating a collection of training instances, when rater availability is constant, consisted of the following steps:

1. **Select instances:** None of the datasets used imposed any order on the instances they contained. However, in conditions of dynamic rater availability, instances would have been picked in a certain sequence, and then presented to the raters. We used two approaches to determine this sequence: random presentation, and an approach based on active learning.

   Following other work in the area of dynamic estimation of rater reliability [55, 205], active learning was used to determine the order of presentation of the instances in the VAM datasets. Active learning [158] is a semi-supervised machine learning approach that can be used to build accurate classifiers and predictors from collections of unrated data with minimal rating effort. This is achieved by only rating those instances from a large pool that are deemed to be *most informative* by some selection strategy.

   At each iteration of our simulated rating process we used active learning to select the five most informative instances from those that had not yet been rated. Active learning was used solely for the purpose of selecting the sequence of instances to be presented to the raters. The uncertainty sampling active learning approach of Burbidge et al. [25] was used: it requires training an ensemble of predictors on non-overlapping training sets. Then the informativeness of a candidate instance from the pool is measured as the difference in the predictions given by ensemble members. In the very beginning of the rating process, we selected 10 instances, randomly distributed across ensemble members. The selection was performed using a deterministic clustering approach [85]. As new instances were rated, they were added to the training sets of ensemble members, which were then re-trained. Support Vector Regression (SVR) predictors were used in the ensemble, as according to our comparison of machine learning techniques, kernel methods perform significantly better than a number of other algorithms, when applied to natural emotional speech (for details see Appendix A). Occasionally, two instances had the same informativeness score. Such ties were resolved in random order.

For some datasets, we could not use active learning to determine the order of presentation, as features were not supplied with these datasets, making the use of active learning impossible. For these datasets, we used a random order of presentation.

2. **Select raters:** At each step the N top raters with the highest reliability score were chosen, and their ratings were used for the instances chosen by the previous phase. Initially, the reliability scores of all raters were the same, because nothing was known about their performance, and the selection of raters was random. As instances were rated, reliabilities were updated, and reliable raters were discovered. Ties were resolved in random order.

3. **Calculate the predicted rating:** We used the average of the N ratings received for each instance as the predicted rating for that instance[1].

4. **Update the rater reliabilities:** The closer the rating given by a rater was to the predicted rating for an instance, the more reliable the rater was deemed to be. The reward for a rater was proportional to the difference between the predicted rating and the rating provided by the rater, and was calculated as follows[2]:

$$reward = 1 - \frac{|prediction - rating|}{max\_val - min\_val}, \tag{5.1}$$

where *prediction* is the prediction calculated at Step 3, *rating* is the rating supplied by a rater, and *max_val* and *min_val* are the maximum and minimum values on the rating scale respectively. If the rater provided a correct rating ($prediction = rating$), the numerator in the reward function is 0, and so the reward is 1 (the maximum value of the reward). Similarly, if the difference between *prediction* and *rating* is maximal (i.e. the full width of the rating scale), the reward is 0. Thus, the reward is bounded in the $[0, 1]$ interval. All rewards received by raters were stored and used to select the most reliable raters.

[1] It might be surprising that the unweighted average is used, as it does not take into account differences in rater reliability. However, our preliminary experiments revealed that weighting the average by rater reliabilities did not have a big impact as top N raters usually have very similar reliability. As a result, the difference between weighted average and unweighted average was small.

[2] A similar formula can also be used in the binary or multi-class classification. In a binary classification scenario the reward can be equal to 1, if *rating* is the same as *prediction* [55]. Multi-class classification can use a majority vote of all ratings received for the instance as the prediction value.

This formulation of the reward sought the subset of raters who give most similar ratings and assumed that these raters will also be the most reliable ones. State-of-the-art algorithms for determining predictions in a multiple rater scenario rely on this assumption [55, 136], although it is rarely stated explicitly. Rewards were calculated in the same way across all approaches used in this chapter. The way in which these rewards were handled, however, is where the differences between multi-armed bandit approaches manifested themselves.

5. **Go to Step 1, if there are any unrated training instances.**

5.2 Evaluation

This experiment investigates whether multi-armed bandit approaches to rater selection lead to predictions of better quality than those received using the IEThresh method (considered to be a state-of-the-art approach), and a naïve approach, in which raters were selected randomly. This experiment used the *BoredomVideos*, *ImageWordSimilarity* and three VAM datasets (*VAM_Activation*, *VAM_Evaluation* and *VAM_Power*). The following dynamic rater reliability estimation approaches were compared in this experiment:

1. **Random (baseline):** no estimation of rater reliability takes place, and N random raters are picked at Step #2 of the approach described in Section 5.1.

2. **Best Overall (baseline):** before the rating process starts, the algorithm by Raykar et al. [136] is used to calculate the reliabilities of all raters using the full set of ratings available in our datasets. At Step #2 only the top N raters are asked to provide ratings. This approach is an artificial baseline, and represents the best accuracy achievable using N raters. This baseline behaves as if rater reliabilities were known beforehand.

3. **IEThresh (baseline):** this upper-confidence bound technique (described in Section 3.6) is used to calculate the rater reliability at Step #4 of the approach. Due to a lack of information in the literature on which to base the value of ϵ, we ran all

experiments using IEThresh with values of $\epsilon = 0.1, 0.2, ..., 0.9$ and reported results using the value that led to the lowest MAHP value.

4. **ϵ-first:** one of the simplest multi-armed bandit algorithms that divides all the rating process into exploration and exploitation phases (Section 3.5.2), the sizes of which are determined by the ϵ parameter. Similar to the work of Vermorel and Mohri [187], we experimented with $\epsilon = 0.05, 0.10, 0.15$. It turned out that different values of ϵ did not exhibit significant differences in our experiments, so only the results for $\epsilon = 0.10$ are reported.

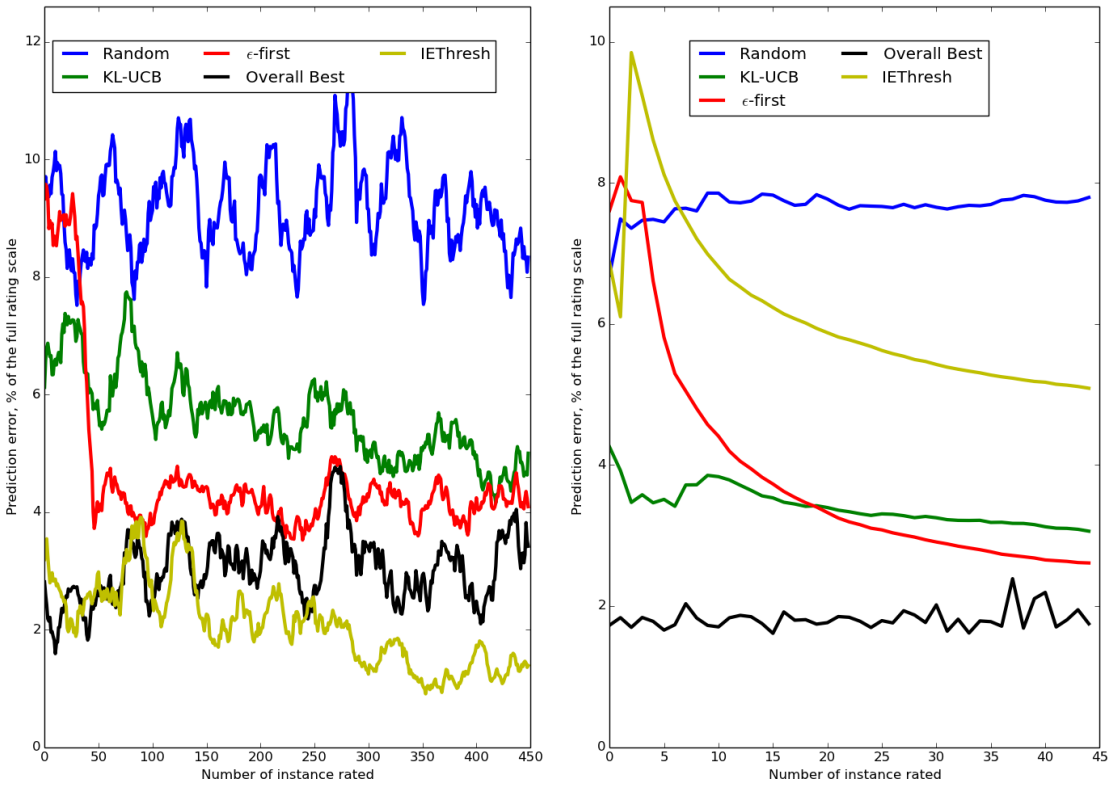
5. **KL-UCB:** an advanced upper-confidence bound technique that does not require any parameters and uses a formula 3.1 to calculate rater reliability.

For the VAM datasets we have performed experiments for $N = 3, 5, 7, 9$. As the total number of raters in *ImageWordSimilarity* and *BoredomVideos* datasets is lower, we used $N = 3, 4, 5, 6$ for these. In IEThresh the number of raters used at each step is determined by the threshold reliability score.

Average absolute error and cost (aggregated as described in Section 4.2 with $w_c = w_e = 0.5$) were the performance measures used in this experiment (as the constant rater availability was considered, T was always equal to zero and so was not included into the performance measure). To account for the random selection associated with ties in the active learning process when selecting instances in the VAM datasets, we report average errors across 5 different runs. For those rating algorithms that contained a random component, Random and ϵ-first, we ran each rating experiment on each instance sequence ten times using different random seeds and reported average errors. In the experiments using the *ImageWordSimilarity* and *BoredomVideos* datasets there was an additional random component associated with the selection of instances, as the order of instance presentation was random. In order to compensate for this we reported averages of 100 runs of the experiment for these two datasets (in each run instances were presented in a different order).

5.3 Results

The general behaviour of all algorithms is shown in Figure 5.1, using as an illustrative example the results of experiments on the *VAM_Power* and *BoredomVideos* datasets with $N = 7$ and $N = 6$. The number of instances rated is given on the horizontal axis, while the vertical axis gives the error of the prediction for this particular instance expressed as a percentage of the full rating scale. Each line represents how the error of prediction changed over time.



(a) *VAM_Power*, N = 7 (IEThresh uses 17 raters on average, $\epsilon = 0.8$)

(b) *BoredomVideos*, N = 6 (IEThresh uses 8.85 raters on average, $\epsilon = 0.7$)

Figure 5.1: Results of experiments with constant reliability. As active learning was used in VAM datasets, the sequence of instances presented for rating was very similar from run to run. This resulted in spiky curves, so for the purpose of illustration, we plotted a moving average of twenty values instead of the original error values. In *BoredomVideos* and *ImageWordSimilarity* the sequence of instances varied a lot, so the error curves are smoother.

The performance level achieved using the Random and Overall-Best approaches did not change much during the process as these algorithms do not take reliability into account.

In contrast, ϵ-first, KL-UCB and IEThresh learn about noisy raters during the process. That allows them to achieve lower and lower error as the rating process progresses. During its exploration stage, ϵ-first picks random raters, so its performance is similar to Random. However, when exploitation starts, the error given by ϵ-first drops quickly as it starts to take rater reliability into account. KL-UCB, as a UCB-algorithm, does not have distinct phases, so its error decreases more slowly. The same behaviour is also seen with IEThresh. This can be explained by the dynamic choice of N that IEThresh uses. IEThresh performs well at the start, because all raters are asked to rate (i.e. none of them have rated yet, so they all have equally high reliability). Later, when rater reliabilities are estimated more precisely, N decreases. The good performance in error for IEThresh has a high associated cost with high numbers of ratings needed.

Table 5.1 shows the prediction errors achieved by the different algorithms, expressed as a percentage of the whole rating scale, as well as costs measured as the total number of ratings collected (the rows for IEThresh also show the value of the ϵ parameter that achieved the reported performance which was the best possible based on MAHP).

In order to compare the performance of algorithms, we aggregated cost and error as described in Section 4.2. For each dataset, we had a single result for IEThresh, but four different results corresponding to different values of N for all other approaches. Among those four we selected for comparison the value of N that led to the lowest MAHP measure value. In *BoredomVideos* all algorithms achieved the lowest MAHP metric value when $N = 6$, for all other datasets it happened when $N = 3$. The values of the aggregated MAHP measure and ranks of approaches are presented in Table 5.2. As expected, the Overall-Best algorithm is always the best, while Random always comes second to last. Both KL-UCB and ϵ-first were always worse than Overall-Best, but better than the other approaches. The average rank of IEThresh is the lowest as this algorithm uses a large number of raters which results in high cost but not necessarily lower error than the other approaches. In the VAM datasets IEThresh used almost a half of the overall rater population. On *BoredomVideos* and *ImageWordSimilarity* it was even more dramatic: almost all raters were asked to rate every instance. However, using a large number of raters did not lead

Table 5.1: Results of experiments with constant availability. The table lists errors and costs (measured as the total number of ratings collected) for all experiments and all approaches. The errors are given as a percentage of the full rating scale. For IEThresh the average value of N during the process is reported. All errors are reported with 95% confidence intervals, as is cost for IEThresh. Cost for other approaches was determined in advance, and, therefore, remained same across all runs.

Dataset	N	ϵ-first		KL-UCB		Random		Overall-Best		IEThresh	
		Error	Cost	Error	Cost	Error	Cost	Error	Cost	Error	Cost
BoredomVideos	3	11.12±0.89	135	8.55±0.28	135	15.98±0.14	135	3.99±0.00	135	-	-
BoredomVideos	4	7.43±0.76	180	5.83±0.16	180	12.87±0.16	180	3.63±0.00	180	-	-
BoredomVideos	5	5.55±0.6	225	4.51±0.08	225	9.76±0.08	225	4.24±0.00	225	-	-
BoredomVideos	6	3.86±0.46	270	3.4±0.08	270	7.67±0.06	270	1.82±0.00	270	-	-
BoredomVideos	8.85 ($\epsilon = 0.7$)	-	-	-	-	-	-	-	-	6.24±0.34	437±21
ImageWordSimilarity	3	10.18±0.25	249	10.61±0.28	249	11.87±0.1	249	3.8±0.00	249	-	-
ImageWordSimilarity	4	8.73±0.36	332	8.71±0.16	332	10.06±0.06	332	2.98±0.00	332	-	-
ImageWordSimilarity	5	7.24±0.17	415	7.35±0.11	415	8.37±0.01	415	3.65±0.00	415	-	-
ImageWordSimilarity	6	6.28±0.14	498	6.1±0.06	498	7.64±0.05	498	3.58±0.00	498	-	-
ImageWordSimilarity	9.95 ($\epsilon = 0.3$)	-	-	-	-	-	-	-	-	4.75±0.02	826±2
VAM_Activation	3	6.51±0.34	1404	10.01±0.39	1404	14.62±0.46	1404	4.53±0.00	1404	-	-
VAM_Activation	5	4.73±0.27	2340	7.03±0.27	2340	11.53±0.4	2340	3.38±0.00	2340	-	-
VAM_Activation	7	3.88±0.22	3276	5.32±0.22	3276	9.87±0.32	3276	2.4±0.00	3276	-	-
VAM_Activation	9	2.83±0.22	4212	4.15±0.19	4212	8.61±0.32	4212	1.75±0.00	4212	-	-
VAM_Activation	12.33 ($\epsilon = 0.9$)	-	-	-	-	-	-	-	-	4.74±0.37	5770±108
VAM_Evaluation	3	6.88±0.28	1404	9.46±0.32	1404	11.48±0.37	1404	4.43±0.00	1404	-	-
VAM_Evaluation	5	4.82±0.23	2340	6.58±0.23	2340	9.02±0.29	2340	3.39±0.00	2340	-	-
VAM_Evaluation	7	3.87±0.18	3276	5.07±0.19	3276	7.66±0.25	3276	2.84±0.00	3276	-	-
VAM_Evaluation	9	3.35±0.18	4212	4.08±0.15	4212	6.42±0.22	4212	2.39±0.00	4212	-	-
VAM_Evaluation	16.42 ($\epsilon = 0.9$)	-	-	-	-	-	-	-	-	3.70±0.27	7685±84
VAM_Power	3	7.79±0.33	1404	10.4±0.35	1404	14.17±0.49	1404	4.9±0.00	1404	-	-
VAM_Power	5	5.69±0.27	2340	7.29±0.25	2340	10.95±0.35	2340	3.34±0.00	2340	-	-
VAM_Power	7	4.42±0.22	3276	5.62±0.2	3276	9.13±0.31	3276	3.04±0.00	3276	-	-
VAM_Power	9	3.17±0.19	4212	4.58±0.18	4212	7.81±0.28	4212	2.51±0.00	4212	-	-
VAM_Power	17 ($\epsilon = 0.8$)	-	-	-	-	-	-	-	-	2.68±0.15	7956±51

Table 5.2: Ranking of algorithms according to their MAHP metric values. All approaches except IEThresh use $N = 9$ for the VAM datasets and $N = 6$ for *BoredomVideos* and *ImageWordSimilarity*.

Dataset	ϵ-first		KL-UCB		Random		Overall-Best		IEThresh	
	MAHP	Rank	MAHP	Rank	MAHP	Rank	MAHP	Rank	MAHP	Rank
BoredomVideos	0.17283	3	0.16244	2	0.24363	4	0.11868	1	0.27956	5
ImageWordSimilarity	0.18374	2	0.18758	3	0.1984	4	0.11226	1	0.22859	5
VAM_Activation	0.14094	2	0.17477	3	0.21122	4	0.11757	1	0.24381	5
VAM_Evaluation	0.14213	2	0.16666	3	0.1836	4	0.11405	1	0.24385	5
VAM_Power	0.14205	2	0.16412	3	0.19158	4	0.11266	1	0.19833	5
Average rank	2.2		2.8		4		1		5	

to a large decrease in error, as shown by the MAHP values. Overall, multi-armed bandit approaches in our experiments were always more accurate and cheap than IEThresh or Random.

Our results present strong evidence of the suitability of multi-armed bandit approaches to the task of dynamic estimation of rater reliabilities. Multi-armed bandit approaches proved to be better than IEThresh and Random. However, the superiority of the Overall-Best approach over the multi-armed bandit approaches shows that there is still some room for improvement.

Some of the results shown in Table 5.1 might seem counterintuitive. One might expect that when one more rater is added, the quality of predictions should improve. However, when the fifth rater was added in the Overall-Best approach on the *ImageWordSimilarity* and the *BoredomVideos* datasets, the error increased. Investigation showed that this happened because the additional N-th rater was much less reliable than the previously selected $N - 1$ top raters. Taking ratings from this rater inevitably worsened the quality of predictions. Figure 5.2 shows the reliabilities of all raters on the *ImageWordSimilarity* dataset calculated using the approach by Raykar et al. [136], sorted from the most reliable (rater A) to the least reliable rater (rater J). The top four raters (A-D) were much more accurate than the rest of the raters. Using these four raters resulted in the error of 2.98%, but when a much less reliable rater (E) was added, the error went up to 3.65%. At the same time, we did not see this effect in the multi-armed bandit-based approaches as they

were not always able to pick the top raters (e.g. in the beginning of the rating process, when reliabilities have not been estimated precisely yet). This situation was caused by a small number of raters in these datasets and is unlikely to happen in real-life conditions, where number of raters usually will be much bigger. It should be noted that this problem is likely to disappear if prediction is calculated not as an unweighted average, but as an average weighted by rater reliabilities.

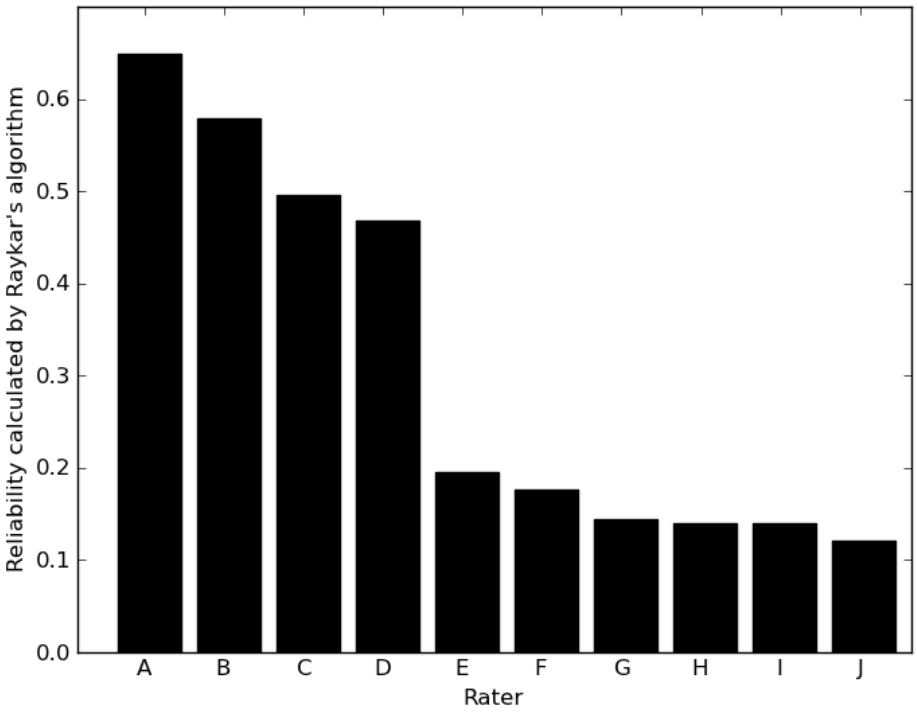


Figure 5.2: Raykar's rater reliabilities on *ImageWordSimilarity* dataset.

5.4 Conclusions

In this chapter we evaluated multi-armed bandit approaches for the task of dynamic estimation of rater reliability. Both of the multi-armed bandit approaches we used—ϵ-first and KL-UCB—led to higher accuracy compared to a random selection of raters. IEThresh, the state-of-the-art baseline, in some cases was able to achieve accuracy similar to ϵ-first and KL-UCB, but required more than twice the number of ratings, and therefore,

was much more expensive. When cost in terms of numbers of ratings and accuracy were combined, IEThresh proved to be the worst approach, ranking below even a random selection of raters. Overall, these results strongly suggest that multi-armed bandits can be successfully used to decrease the error of predictions. However, the superiority of the Overall Best approach suggests that further improvements in multi-armed bandit approaches are possible. In the next chapter we consider one such improvement, namely, getting additional ratings for the instances rated at the very beginning of the process, when rater reliability has not yet been estimated precisely, and, as a result, ratings from noisy raters are sometimes collected for these instances.

Chapter 6

Bootstrap Issue in the Scenario of Constant Rater Availability

This chapter is devoted to the *bootstrap problem* associated with multi-armed bandits used in estimating rater reliability. As these algorithms seek balance between exploration and exploitation, it is possible that instances rated at the initial exploration stage will receive poor ratings. However, at the end of the process, when rater reliabilities are estimated accurately, we can ask reliable raters to provide additional ratings for the "exploration-phase" instances and thus improve the accuracy of predictions.

The goal of the experiments in this chapter is to evaluate several approaches to increasing the accuracy of predictions by getting additional ratings for the instances that have been rated during the exploration phase. Namely, we are interested in whether or not such *re-rating* can increase the quality of predictions. Section 6.1 introduces a set of approaches to detect the boundary between exploration and exploitation, as well as describes how these approaches were evaluated. Section 6.2 discusses the results, while Section 6.3 concludes the chapter.

6.1 Methodology

The experiment for evaluating re-rating approaches consisted of two stages:

1. **Original rating process**, a rating process as described in Chapter 5.

2. **Re-rating** that proceeded as follows:

   (a) Instances for which the acquisition of additional ratings was required were selected. These were the instances rated during the exploration phase.

   (b) Additional ratings for the selected instances were solicited from the N most reliable raters (as determined at the end of the original rating process) to replace those previously collected. However, it is possible that some of these most reliable raters have already provided ratings for some of these instances at the exploration stage. In such situations we simply re-used the old rating. Collecting a new rating would amount to "purchasing" multiple ratings for the same training instance, from the same individual.

   (c) The predictions for the selected instances were updated using the average of the newly acquired ratings.

In this experiment we used the VAM datasets, as well as *Jester* and *MovieLens*. The number of raters asked to rate each training instance was varied as $N = 3, 5, 7, 9, 11, 13, 15$.

Different approaches to re-rating covered in this chapter differed in terms of how Step (a) of the re-rating stage was executed. We used two baselines and two re-rating approaches in our experiments.

We considered the following re-rating approaches:

1. **Fixed:** the first $x\%$ of the training instances rated in the original rating process were selected for re-rating.

2. **Trend-based:** the number of training instances to be re-rated was not set in advance, but was determined via trend analysis. We assumed that there exists a border between exploration and exploitation in the original rating process. When this border was found, only training instances rated at the exploration phase were re-rated.

The baselines were as follows:

1. **None:** no re-rating happened at all (zero training instances were selected for re-rating).

2. **Full:** all training instances rated during the original rating process were re-rated.

As discussed, noisy ratings are often collected during the exploration phase. This means that predictions for the training instances rated at this phase may be unreliable. As the exploration stage progresses, rater reliabilities are learned, and unreliable raters are chosen less and less frequently, thus, the error of predictions becomes lower. In practice, the gold standard is not known during the rating process and so the error in predictions compared to it cannot be calculated. Consequently, this error can not be used in locating the start of the exploitation phase. However, the standard deviation of the N ratings received for each training instance can be used as a proxy measure of error, as reliable raters tend to agree with one another. Our initial experiments revealed that the behaviour of the standard deviation was generally similar to that of the error making it a suitable proxy for the error. As seen in Figure 6.1, the standard deviation (and, therefore, the error in predictions) exhibits a negative trend, when compared to the gold standard for the exploration stage. Finally, when rater reliabilities are estimated well enough, only reliable raters are asked, and the error of predictions remains stable and relatively low.

The process of finding the boundary between the exploration and exploitation phases is illustrated in Figure 6.1. The figure is not based on any of the datasets used in this thesis, it illustrates the ideal hypothetic case. To measure the trend we calculated the sequence of all standard deviations $(\sigma_1, \sigma_2, ..., \sigma_T)$ for ratings received in the original rating process, where T was the total number of training instances. Each σ_i was the standard deviation of the ratings supplied by raters for the training instance rated at the i-th round. The Mann-Kendall test [117] was used to test for a negative trend in this sequence. If there was a negative trend, the first standard deviation was removed and the test for a negative trend was performed again on the remaining sequence $(\sigma_2, \sigma_3, ..., \sigma_T)$. The process continued until the point was found at which the standard deviations in the sequence did not exhibit a negative trend. This was considered to be the point at which the exploration phase

ended, and the exploitation phase began.

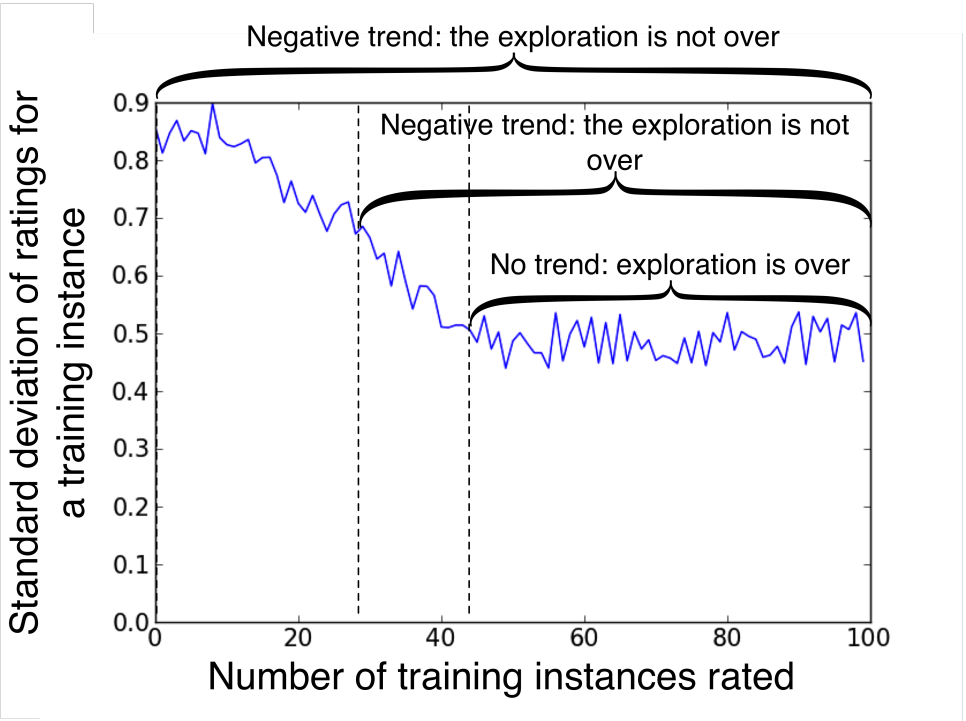


Figure 6.1: Detection of the boundary between exploration and exploitation using trend analysis. In this example, all training instances rated before the training instance #43 were rated at the exploration stage, and had to be re-rated. The figure is not based on any of the datasets used in this thesis, it illustrates the ideal hypothetic case.

In our experiments we tested the re-rating approaches described above, using two multi-armed bandit approaches: ϵ-first and KL-UCB. These were used in the same way as in Chapter 5. Cost and error were aggregated using MAHP, and averaged over a number of runs, as described in Section 5.2.

6.2 Results

The results for the experiments where ϵ-first was used are given in Table 6.1. With ϵ-first, in 30 out of 35 experiments, the Trend-based approach re-rated between 10% and 10.1% of training instances. Therefore, the border between exploration and exploitation usually lay in the region of $\epsilon = 0.1$, so it appeared that the Trend-based approach was just finding the ϵ-first predefined border. Figure 6.2 shows just one standard deviation graph as an illustration, but very similar behaviour was observed in the other experiments as

Table 6.1: Results for re-rating experiments where ϵ-first was used to perform dynamic estimation of rater reliability. Cost is given as the total number of ratings collected, error is measured in the percentage of the whole rating scale.

Dataset	N	None		Full		Fixed, $x = 10$		Trend-based	
		Cost	Error	Cost	Error	Cost	Error	Cost	Error
Jester	3	300	9.49	368	9.09	329	9.2	326	9.2
Jester	5	500	5.94	567	5.56	540	5.61	540	5.65
Jester	7	700	4.73	770	4.37	743	4.48	742	4.48
Jester	9	900	3.9	985	3.72	944	3.72	946	3.72
Jester	11	1100	3.71	1177	3.65	1145	3.59	1148	3.61
Jester	13	1300	3.26	1371	3.29	1340	3.17	1343	3.17
Jester	15	1500	2.97	1552	2.92	1532	2.89	1535	2.89
MovieLens	3	864	6.83	995	6.59	939	6.65	940	6.63
MovieLens	5	1440	4.72	1628	4.48	1548	4.54	1553	4.52
MovieLens	7	2016	3.42	2190	3.28	2143	3.27	2145	3.27
MovieLens	9	2592	2.78	2798	2.58	2724	2.64	2737	2.63
MovieLens	11	3168	2.28	3382	2.09	3311	2.15	3308	2.15
MovieLens	13	3744	2.17	3928	2.1	3876	2.1	3875	2.1
MovieLens	15	4320	1.99	4481	1.98	4420	1.94	4437	1.96
VAM_Activation	3	1404	5.99	1536	5.29	1523	5.35	1528	5.35
VAM_Activation	5	2340	4.49	2535	3.84	2539	3.83	2530	3.83
VAM_Activation	7	3276	3.4	3541	2.89	3511	2.87	3520	2.87
VAM_Activation	9	4212	2.4	4502	1.9	4486	1.9	4492	1.9
VAM_Activation	11	5148	1.97	5447	1.42	5442	1.42	5444	1.42
VAM_Activation	13	6084	1.78	6402	1.21	6394	1.22	6401	1.22
VAM_Activation	15	7020	1.77	7336	1.23	7306	1.27	7322	1.27
VAM_Evaluation	3	1404	6.36	1640	6.26	1534	6.16	1545	6.19
VAM_Evaluation	5	2340	4.52	2554	4.16	2527	4.15	2529	4.15
VAM_Evaluation	7	3276	3.32	3559	3.07	3506	3.04	3510	3.04
VAM_Evaluation	9	4212	2.67	4521	2.44	4493	2.41	4489	2.41
VAM_Evaluation	11	5148	2	5460	1.74	5453	1.74	5447	1.74
VAM_Evaluation	13	6084	1.58	6405	1.29	6385	1.3	6398	1.3
VAM_Evaluation	15	7020	1.23	7337	0.88	7310	0.91	7320	0.9
VAM_Power	3	1404	7.51	1616	7.14	1530	7	1532	7.01
VAM_Power	5	2340	5.04	2681	4.65	2530	4.55	2532	4.55
VAM_Power	7	3276	4.28	3608	3.88	3506	3.9	3511	3.89
VAM_Power	9	4212	3.17	4541	2.84	4488	2.84	4498	2.83
VAM_Power	11	5148	2.73	5468	2.41	5449	2.39	5446	2.39
VAM_Power	13	6084	2.12	6393	1.75	6393	1.76	6388	1.76
VAM_Power	15	7020	1.68	7333	1.27	7331	1.27	7332	1.27

well. Such a crisp distinction between exploration and exploitation was not unexpected, as ϵ-first explicitly explores during first $\epsilon \cdot T$ rounds, and exploits the rest of the time.

A comparison of average ranks of approaches, for the case when ϵ-first was used to estimate rater reliability, shows that there was a statistically significant difference between re-rating approaches (Friedman test p-value <0.001). The following two groups were identified by the Bergmann-Hommel test with significance at the $\alpha = 0.05$ level[1] (average

[1]The results reported below use $\alpha = 0.05$, unless otherwise specified.

ranks are given in parentheses):

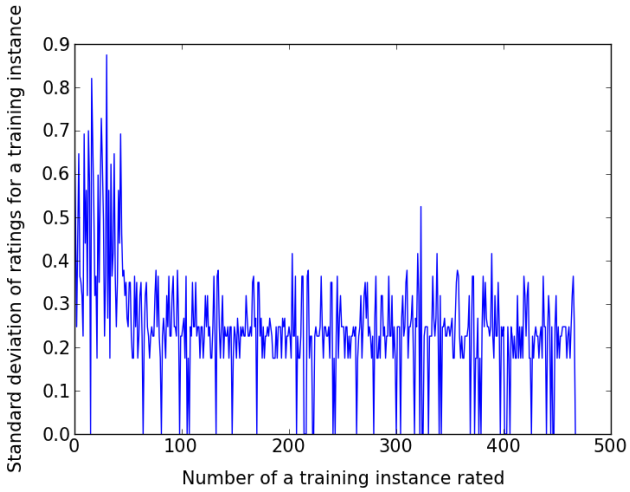


Figure 6.2: Change in standard deviation of ratings, while training instances are being rated (*Activation*, $N = 3$).

1. Fixed, $x = 10$ (1.94) and Trend-based (2.17)
2. None (2.77) and Full (3.11)

None and Full were the worst approaches, while Fixed and Trend-based turned out to be the best. As the boundary between exploration and exploitation was almost always at about 10%, the Fixed approach worked well, always re-rating 10% of training instances. Although there is no statistically significant difference between the Fixed and Trend-based approaches, we would recommend the use of the Fixed approach as a simpler alternative, when ϵ-first is used to select raters dynamically. The x value can easily be set from the the ϵ parameter in ϵ-first. The detailed ranks of approaches are given in Table 6.2.

In our experiments, the first $\epsilon \cdot T$ training instances had an average error of 6.57%, while the rest had an error of 3.34%. When initial training instances were re-rated using fixed re-rating, the error on those instances dropped to 3.44%, i.e. halved.

When KL-UCB was used, the border between exploration and exploitation on the VAM datasets (as detected by the Trend-based approach) lay in quite a wide range from 5% to 50%. This is in contrast to ϵ-first, where this border was almost always around 10%. Typical standard deviation graph for VAM datasets exhibited a negative trend, however, such graph for *MovieLens* and *Jester* did not exhibit any trends or phases (an example is

Table 6.2: Ranks of re-rating approaches when ϵ-first was used to estimate rater reliability dynamically. MAHP measure aggregates cost and time with the same weights ($W_C = W_E = 0.5$).

Dataset	N	None		Full		Fixed, $x = 10$		Trend-based	
		MAHP	Rank	MAHP	Rank	MAHP	Rank	MAHP	Rank
Jester	3	0.2412	1	0.2615	4	0.2487	3	0.2476	2
Jester	5	0.2465	1	0.2540	4	0.2490	2	0.2499	3
Jester	7	0.2491	1	0.2511	4	0.2497	3	0.2496	2
Jester	9	0.2485	1	0.2539	4	0.2485	2	0.2488	3
Jester	11	0.2477	1	0.2541	4	0.2485	2	0.2496	3
Jester	13	0.2478	1	0.2557	4	0.2481	2	0.2484	3
Jester	15	0.2498	3	0.2519	4	0.2490	1	0.2492	2
MovieLens	3	0.2432	1	0.2563	4	0.2501	3	0.2499	2
MovieLens	5	0.2456	1	0.2545	4	0.2498	3	0.2496	2
MovieLens	7	0.2476	1	0.2527	4	0.2496	2	0.2497	3
MovieLens	9	0.2499	3	0.2502	4	0.2497	1	0.2498	2
MovieLens	11	0.2515	4	0.2488	1	0.2497	3	0.2496	2
MovieLens	13	0.2494	1	0.2513	4	0.2496	3	0.2496	2
MovieLens	15	0.2487	2	0.2527	4	0.2484	1	0.2502	3
VAM_Activation	3	0.2527	4	0.2484	1	0.2488	2	0.2492	3
VAM_Activation	5	0.2571	4	0.2474	3	0.2473	2	0.2469	1
VAM_Activation	7	0.2586	4	0.2478	3	0.2459	1	0.2463	2
VAM_Activation	9	0.2656	4	0.2443	3	0.2439	1	0.2440	2
VAM_Activation	11	0.2753	4	0.2404	3	0.2403	1	0.2403	2
VAM_Activation	13	0.2809	4	0.2375	1	0.2384	2	0.2385	3
VAM_Activation	15	0.2782	4	0.2371	1	0.2404	2	0.2406	3
VAM_Evaluation	3	0.2417	1	0.2591	4	0.2486	2	0.2501	3
VAM_Evaluation	5	0.2502	3	0.2508	4	0.2491	1	0.2492	2
VAM_Evaluation	7	0.2509	3	0.2515	4	0.2484	1	0.2486	2
VAM_Evaluation	9	0.2528	4	0.2504	3	0.2481	2	0.2480	1
VAM_Evaluation	11	0.2575	4	0.2473	3	0.2472	2	0.2470	1
VAM_Evaluation	13	0.2637	4	0.2445	1	0.2450	2	0.2453	3
VAM_Evaluation	15	0.2757	4	0.2384	1	0.2420	3	0.2408	2
VAM_Power	3	0.2459	1	0.2573	4	0.2479	2	0.2482	3
VAM_Power	5	0.2495	3	0.2565	4	0.2465	1	0.2466	2
VAM_Power	7	0.2515	4	0.2513	3	0.2483	2	0.2482	1
VAM_Power	9	0.2539	4	0.2495	3	0.2480	2	0.2479	1
VAM_Power	11	0.2566	4	0.2485	3	0.2470	2	0.2470	1
VAM_Power	13	0.2629	4	0.2448	1	0.2455	3	0.2454	2
VAM_Power	15	0.2721	4	0.2418	3	0.2418	1	0.2418	2
Average rank		2.77		3.11		1.94		2.17	

given on Figure 6.3). One possible explanation is that Jester and MovieLens represent more subjective problems, compared to the VAM datasets. In such more subjective problems it was more difficult to find a subset of raters tending to agree with one another.

In order to investigate this, we calculated average absolute errors[2] for all raters in all five datasets. In Figure 6.4 red crosses represent noisy, artificially generated raters[3]

[2]Error was calculated as an average absolute difference between the gold standard and ratings provided by the rater.

[3]The generation of noisy raters is described in detail in Section 4.1.

who do not agree with each other, as their ratings were generated independently of one another. The errors of raters who were in the original datasets (blue crosses) are spread uniformly in all datasets, but the range of rater errors is bigger on the *MovieLens* and *Jester* datasets than on the VAM datasets. This means that, in general, raters on the *MovieLens* and *Jester* datasets tended to disagree with each other more than raters on the VAM datasets. Thus, it was more difficult for KL-UCB to pick a set of raters who agree, which explains the absence of any trend in the standard deviation graphs.

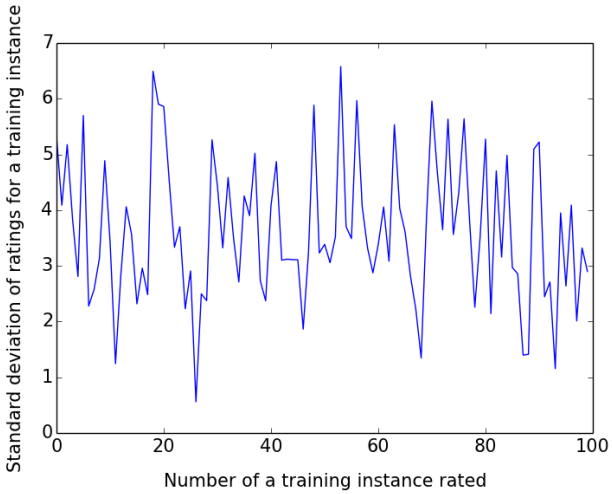


Figure 6.3: Change in standard deviation of ratings, as training instances are being rated (*Jester*, $N = 5$).

The detailed results of the experiments with KL-UCB are given in Table 6.3. First we launched the fixed approach with $x = 50$ in order to ensure that it always re-rates all exploration-phase training instances. The comparison of ranks for this case is given in Table 6.4. Approaches were split into three groups (Friedman p-value <0.001):

1. None (1.8), Trend-based (2.03)
2. Fixed, $x = 50$ (2.69)
3. Full (3.51)

The trend-based approach often re-rated only a few initial training instances and therefore did not produce a big difference in cost or error, compared to no re-rating. When the fixed approach was used, the average error over the initial 50% of training instances changed from 5.54% (no re-rating) to 4.67%. However, this decrease in error required sig-

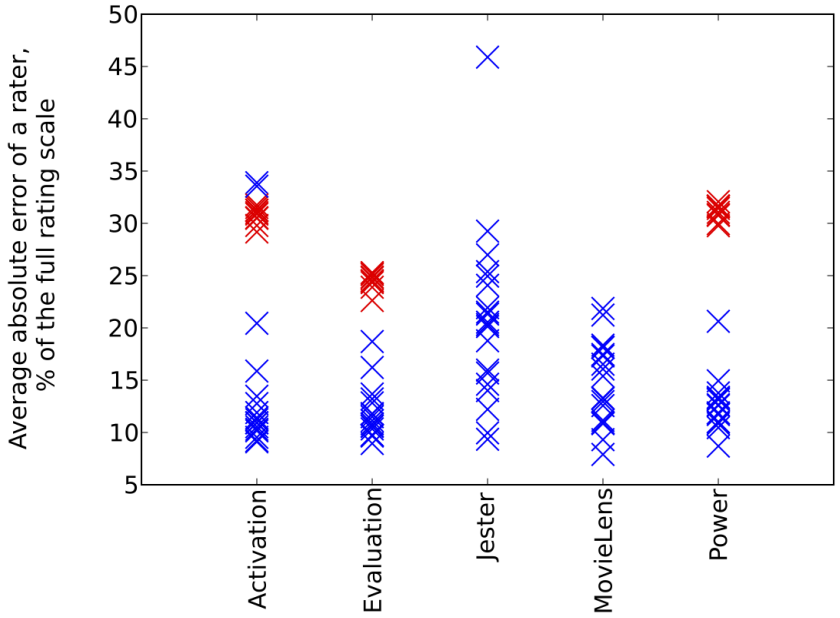


Figure 6.4: Mean errors of raters. Red crosses represent artificially generated, noisy raters, while blue crosses correspond to raters originally present in datasets. It is difficult to find a subset of raters who agree with each other in *MovieLens* and *Jester*.

nificant additional costs: compared to an average $C = 3240.8$ for no re-rating, the Fixed approach resulted in $C = 4021.23$ (an increase of 24.1%). This poor "value for money" resulted in the Fixed approach ranking worse than no re-rating.

The grouping did not change when we used $x = 25$ instead of 50 in the Fixed approach (Friedman p-value <0.001):

1. None (1.83), Trend-based (1.97)
2. Fixed, $x = 25$ (2.69)
3. Full (3.51)

The details of ranks for each experiment are given in Table 6.5. In the Fixed approach, the significant additional cost resulted in a 1.04% decrease in error for the initial 25% of training instances, from 5.75% to 4.71%. We also tried several different values of x in the fixed approach, but the grouping remained the same.

The overall result is that the fixed re-rating approach proved to be beneficial in the case when ϵ-first is used to estimate rater reliability dynamically. In KL-UCB our results do not suggest that re-rating, as proposed in this chapter, should be recommended.

Table 6.3: Results for re-rating experiments where KL-UCB was used to perform dynamic estimation of rater reliability. Cost is given as the total number of ratings collected, error is measured in the percentage of the whole rating scale.

Dataset	N	None		Full		Fixed, $x = 25$		Fixed, $x = 50$		Trend-based	
		Cost	Error	Cost	Error	Cost	Error	Cost	Error	Cost	Error
Jester	3	317	9.81	561	9.4	373	9.78	435	9.5	319	9.86
Jester	5	515	6.88	853	6.74	590	6.85	678	6.8	516	6.93
Jester	7	713	5.46	1069	5.25	827	5.36	935	5.28	716	5.48
Jester	9	911	4.35	1269	4.3	1011	4.28	1095	4.25	915	4.36
Jester	11	1109	3.72	1432	3.66	1206	3.69	1281	3.63	1110	3.73
Jester	13	1307	3.32	1598	3.34	1385	3.29	1467	3.3	1310	3.31
Jester	15	1505	2.9	1737	2.86	1560	2.86	1622	2.85	1507	2.89
MovieLens	3	881	7.61	1588	7.51	1065	7.58	1249	7.54	885	7.61
MovieLens	5	1455	5.43	2441	5.04	1734	5.27	2028	5.18	1461	5.43
MovieLens	7	2029	4.17	3079	3.87	2313	4.04	2565	3.97	2033	4.17
MovieLens	9	2603	3.33	3610	3.21	2918	3.26	3239	3.2	2630	3.32
MovieLens	11	3177	2.78	4076	2.62	3439	2.7	3657	2.65	3188	2.77
MovieLens	13	3751	2.32	4538	2.12	4005	2.23	4241	2.17	3756	2.31
MovieLens	15	4325	2	5003	1.94	4547	1.95	4737	1.95	4345	2
VAM_Activation	3	1428	11.02	2671	11.94	1738	11.09	2049	11.32	1444	11.06
VAM_Activation	5	2362	8.04	4247	8.26	2827	7.78	3296	7.75	2481	7.97
VAM_Activation	7	3296	6.48	5484	5.08	3876	5.88	4447	5.45	3520	6.22
VAM_Activation	9	4230	5.22	6371	2.23	4858	4.24	5427	3.46	4644	4.49
VAM_Activation	11	5164	4.47	7434	3.05	5816	3.82	6396	3.47	5520	3.96
VAM_Activation	13	6098	3.9	8106	2.92	6725	3.37	7202	3.09	6559	3.45
VAM_Activation	15	7032	3.6	8711	2.66	7601	3.13	8025	2.89	7483	3.28
VAM_Evaluation	3	1428	9.15	2665	8.79	1736	9.08	2043	8.94	1431	9.18
VAM_Evaluation	5	2362	6.67	4236	6.69	2834	6.61	3307	6.55	2432	6.65
VAM_Evaluation	7	3296	5.53	5690	5.8	3877	5.52	4449	5.55	3627	5.54
VAM_Evaluation	9	4230	4.7	6773	3.74	4906	4.35	5571	4.09	4794	4.18
VAM_Evaluation	11	5164	3.92	7631	2.45	5836	3.45	6443	3.08	5264	3.85
VAM_Evaluation	13	6098	3.31	8568	2.57	6815	3.04	7471	2.88	6112	3.31
VAM_Evaluation	15	7032	2.87	9250	2.34	7731	2.56	8398	2.44	7590	2.71
VAM_Power	3	1428	10.77	2658	9.53	1734	10.32	2042	9.88	1490	10.54
VAM_Power	5	2362	7.62	4217	7.08	2839	7.48	3315	7.27	2412	7.57
VAM_Power	7	3296	6.09	5538	4.77	3887	5.61	4475	5.22	3622	5.85
VAM_Power	9	4230	5.14	6611	3.42	4884	4.56	5516	4.09	4555	4.75
VAM_Power	11	5164	4.44	7542	2.55	5799	3.78	6365	3.32	5627	3.96
VAM_Power	13	6098	3.76	8280	2.7	6693	3.32	7195	3	6489	3.36
VAM_Power	15	7032	3.41	8932	2.58	7613	2.99	8082	2.84	7312	3.21

6.3 Conclusions

In this chapter we introduced and evaluated a number of approaches to improving the accuracy of predictions, by re-rating instances rated during the exploration stage of the rating process. The ratings collected for these instances are probably inaccurate, because during the exploration stage, rater reliability has not yet been precisely estimated. At the end of the process, the border between exploration and exploitation was determined, and then the most reliable N raters were asked to provide ratings for the initial instances.

Table 6.4: Ranks of re-rating approaches when KL-UCB was used to estimate rater reliability dynamically. The Fixed approach re-rates 50% of instances. MAHP measure aggregates cost and time with the same weights ($W_C = W_E = 0.5$).

Dataset	N	None		Full		Fixed, $x = 50$		Trend-based	
		MAHP	Rank	MAHP	Rank	MAHP	Rank	MAHP	Rank
Jester	3	0.2223	1	0.2894	4	0.2562	3	0.2235	2
Jester	5	0.2249	1	0.2864	4	0.2565	3	0.2259	2
Jester	7	0.2298	1	0.2759	4	0.2588	3	0.2307	2
Jester	9	0.2341	1	0.2747	4	0.2537	3	0.2349	2
Jester	11	0.2382	1	0.2685	4	0.2529	3	0.2386	2
Jester	13	0.2399	2	0.2661	4	0.2534	3	0.2398	1
Jester	15	0.2441	2	0.2604	4	0.2512	3	0.2438	1
MovieLens	3	0.2194	1	0.2926	4	0.2600	3	0.2199	2
MovieLens	5	0.2253	1	0.2811	4	0.2598	3	0.2257	2
MovieLens	7	0.2321	1	0.2755	4	0.2546	3	0.2323	2
MovieLens	9	0.2344	1	0.2710	4	0.2563	3	0.2352	2
MovieLens	11	0.2406	2	0.2646	4	0.2521	3	0.2406	1
MovieLens	13	0.2448	2	0.2573	4	0.2517	3	0.2444	1
MovieLens	15	0.2440	1	0.2585	4	0.2522	3	0.2446	2
VAM_Activation	3	0.2138	1	0.3044	4	0.2596	3	0.2154	2
VAM_Activation	5	0.2188	1	0.2974	4	0.2538	3	0.2233	2
VAM_Activation	7	0.2343	1	0.2676	4	0.2496	3	0.2372	2
VAM_Activation	9	0.2634	4	0.2113	1	0.2429	2	0.2559	3
VAM_Activation	11	0.2510	4	0.2487	3	0.2461	2	0.2442	1
VAM_Activation	13	0.2523	4	0.2517	3	0.2441	1	0.2461	2
VAM_Activation	15	0.2553	4	0.2442	1	0.2443	2	0.2514	3
VAM_Evaluation	3	0.2188	1	0.2930	4	0.2587	3	0.2194	2
VAM_Evaluation	5	0.2193	1	0.2941	4	0.2571	3	0.2222	2
VAM_Evaluation	7	0.2183	1	0.2937	4	0.2541	3	0.2292	2
VAM_Evaluation	9	0.2360	1	0.2664	4	0.2526	3	0.2369	2
VAM_Evaluation	11	0.2492	3	0.2395	1	0.2468	2	0.2494	4
VAM_Evaluation	13	0.2433	1	0.2541	4	0.2512	3	0.2436	2
VAM_Evaluation	15	0.2457	1	0.2544	4	0.2476	2	0.2480	3
VAM_Power	3	0.2227	1	0.2858	4	0.2550	3	0.2250	2
VAM_Power	5	0.2225	1	0.2866	4	0.2575	3	0.2241	2
VAM_Power	7	0.2325	1	0.2667	4	0.2508	3	0.2389	2
VAM_Power	9	0.2444	2	0.2493	4	0.2490	3	0.2438	1
VAM_Power	11	0.2551	4	0.2336	1	0.2449	2	0.2514	3
VAM_Power	13	0.2525	4	0.2493	3	0.2449	1	0.2462	2
VAM_Power	15	0.2520	4	0.2471	2	0.2466	1	0.2493	3
Average rank		1.8		3.51		2.69		2.03	

These ratings were used to re-calculate the predictions for the initial instances. Our experiments show that re-rating can indeed increase the accuracy of predictions in crowdsourced rating of training sets when multi-armed bandits are used to dynamically estimate rater reliability. For ϵ-first, both Trend-based and Fixed re-rating performed well, but we would recommend using the latter as a simpler alternative. In KL-UCB re-rating approaches did not prove to be significantly better than no rerating at all.

All the experiments conducted in Chapters 5 and 6 considered a scenario of constant

Table 6.5: Ranks of re-rating approaches when KL-UCB was used to estimate rater reliability dynamically. The Fixed approach re-rates 25% of instances. MAHP measure aggregates cost and time with the same weights ($W_C = W_E = 0.5$).

Dataset	N	None		Full		Fixed, $x = 25$		Trend-based	
		MAHP	Rank	MAHP	Rank	MAHP	Rank	MAHP	Rank
Jester	3	0.2258	1	0.2940	4	0.2446	3	0.2271	2
Jester	5	0.2286	1	0.2912	4	0.2442	3	0.2297	2
Jester	7	0.2331	1	0.2799	4	0.2487	3	0.2340	2
Jester	9	0.2363	1	0.2772	4	0.2469	3	0.2371	2
Jester	11	0.2396	1	0.2700	4	0.2488	3	0.2400	2
Jester	13	0.2417	2	0.2681	4	0.2477	3	0.2416	1
Jester	15	0.2452	2	0.2616	4	0.2479	3	0.2449	1
MovieLens	3	0.2237	1	0.2984	4	0.2455	3	0.2242	2
MovieLens	5	0.2294	1	0.2863	4	0.2467	3	0.2299	2
MovieLens	7	0.2347	1	0.2785	4	0.2466	3	0.2349	2
MovieLens	9	0.2370	1	0.2740	4	0.2483	3	0.2379	2
MovieLens	11	0.2419	2	0.2660	4	0.2481	3	0.2419	1
MovieLens	13	0.2457	2	0.2584	4	0.2489	3	0.2454	1
MovieLens	15	0.2453	1	0.2598	4	0.2484	3	0.2459	2
VAM_Activation	3	0.2189	1	0.3116	4	0.2422	3	0.2205	2
VAM_Activation	5	0.2230	1	0.3031	4	0.2400	3	0.2275	2
VAM_Activation	7	0.2362	1	0.2698	4	0.2440	3	0.2392	2
VAM_Activation	9	0.2605	4	0.2090	1	0.2516	2	0.2532	3
VAM_Activation	11	0.2511	4	0.2488	3	0.2463	2	0.2443	1
VAM_Activation	13	0.2519	4	0.2513	3	0.2459	2	0.2457	1
VAM_Activation	15	0.2546	4	0.2436	1	0.2468	2	0.2507	3
VAM_Evaluation	3	0.2230	1	0.2986	4	0.2449	3	0.2236	2
VAM_Evaluation	5	0.2233	1	0.2996	4	0.2435	3	0.2263	2
VAM_Evaluation	7	0.2222	1	0.2990	4	0.2408	3	0.2333	2
VAM_Evaluation	9	0.2379	1	0.2685	4	0.2465	3	0.2388	2
VAM_Evaluation	11	0.2489	3	0.2392	1	0.2483	2	0.2491	4
VAM_Evaluation	13	0.2446	1	0.2554	4	0.2478	3	0.2448	2
VAM_Evaluation	15	0.2469	2	0.2556	4	0.2445	1	0.2492	3
VAM_Power	3	0.2261	1	0.2902	4	0.2439	3	0.2285	2
VAM_Power	5	0.2261	1	0.2913	4	0.2456	3	0.2278	2
VAM_Power	7	0.2346	1	0.2691	4	0.2445	3	0.2410	2
VAM_Power	9	0.2449	2	0.2498	4	0.2479	3	0.2443	1
VAM_Power	11	0.2540	4	0.2326	1	0.2483	2	0.2504	3
VAM_Power	13	0.2516	4	0.2485	3	0.2477	2	0.2454	1
VAM_Power	15	0.2524	4	0.2474	2	0.2459	1	0.2497	3
Average rank		1.83		3.51		2.69		1.97	

rater reliability. Although there are some tasks using this scenario, it is much more common that raters enter and exit the rating process at arbitrary times. In Chapter 7 we cover the use of multi-armed bandits for rater reliability estimation under these more realistic conditions.

Chapter 7

Dynamic Estimation of Rater Reliability in the Scenario of Intermittent Rater Availability

Chapter 6 was concerned with using multi-armed bandits for the estimation of rater reliability in simplified conditions. Namely, we assumed that any rater is constantly engaged in the rating process and, therefore, can instantly provide a rating once asked. Such constant availability, however, is not achievable in many real-life scenarios. For instance, workers on Amazon Mechanical Turk are distributed across many time zones, which can make waiting for a rating from a particular rater infeasible. Also, raters can enter and leave the rating process at any stage without a prior warning. Clearly, the multi-armed bandit approach from the previous chapter has to be adapted to such conditions of intermittent rater availability for it to be deployable in these scenarios.

In general, there are two approaches to dynamic estimation of rater reliability: instance-driven and rater-driven. The first approach consists of picking an instance, collecting all necessary ratings for it and then proceeding to the next instance. As we pointed out in Section 3.3, this can lead to significant delays from the point of view of raters, as they often have to wait for a next instance to become available. The rater-driven approach works in a different way. As soon as a rater becomes available to provide ratings, an instance is specifically picked for that rater. We have developed a novel rater-driven approach to

handle intermittent rater reliability. Our approach is called Dynamic Estimation of Rater Reliability for Regression (DER^3). In this chapter we describe this approach and present an evaluation of it using simulated experiments similar to the ones used in Chapters 5 and 6 (in Chapter 8 we will demonstrate how the technique performs in a real-life deployment). In addition to cost and error, that has been considered so far, in this chapter we also use time to measure the performance of different approaches to measuring rater reliability as now there is a delay between asking a rater and getting a rating. We not only want to get low error for a low cost, but also to gather predictions in as short time as possible.

This chapter is structured as follows. Section 7.1 introduces the DER^3 approach. Section 7.2 describes different ways how multi-armed bandit algorithm were used, while Section 7.3 presents the evaluation results. Section 7.4 concludes the chapter.

7.1 The DER^3 Approach

DER^3 is a multi-armed bandit based dynamic approach that is rater-driven. We wait for a rater to appear and then decide whether to give him instances to rate or to inform him that no instances are currently available and ask him to come back later. The latter happens in one of two conditions: (i) this rater has rated all instances available for rating, (ii) there are some instances unrated by this rater, but these have been rated by other raters who are more reliable than this rater. Thus, unreliable raters are not even asked to provide a rating and, therefore, are not paid.

A flowchart of the DER^3 approach is given in Figure 7.1. DER^3 divides the rating process into two stages: an optional stage of exploration and a stage of exploitation. The instances from the entire corpus are arranged in a sequence which is divided into two non-overlapping groups to be rated at each stage (*exploration_instances* and *exploitation_instances*). When exploration takes place, ratings are accepted from any raters without considering their reliability. As soon as N ratings are received for all *exploration_instances*, the rater reliabilities are estimated for the first time and exploitation begins.

During the exploitation stage ratings are not always accepted from raters when they become available. A rating from a rater is accepted only when there is a good chance that this rating will improve the current prediction for a certain instance. This is performed in the following way. When a rater becomes available, all instances that he has not rated yet are selected. Then, for each of them the median rater reliability is calculated by taking a median of reliability of all raters who rated that instance. The rater's ratings are likely to improve the predictions for those instances where the median rater reliability is lower than the current reliability of the rater. However, it is also possible that the median rater reliability for all instances is higher that the rater's reliability. In such case the rater is not invited to rate anything, and a message inviting him to come back later is displayed.

It should be noted that if a rater is rejected at some stage due to his low reliability and, therefore, inability to improve the current prediction for any of the instances. However, as median reliabilities change during the rating process, it is possible that he can be offered to rate at some stage in the future.

The rating process normally finishes when all instances in *exploitation_instances* have received N ratings. However, sometimes the process has to be stopped sooner. For example, consider an instance that was rated by several very reliable raters. It is possible that none of the other raters has a higher reliability than these raters, thus, none of the other raters will actually be invited to rate it. If at some stage such a situation occurs for all instances that have less than N ratings, the rating process stops.

It should be pointed out that when a rater becomes available a single instance is being selected for rating. If the rater is still available after rating the selected instance, the algorithm starts again[1]. If the rater is available for a certain time to rate a number of instances, the algorithm treats it as a sequence of single "rater is available" events.

At any moment of time there can be numerous instances available for a particular rater: some will be rated by N raters, while some might not be rated at all. We first present instances having zero or just one rating in order to get every instance in the set to be rated by any two raters without considering their reliability. Two is a minimum number

[1]In real-life implementation of this algorithm a track of who rated what should be kept. One of possible solutions is to enter all ratings into a database.

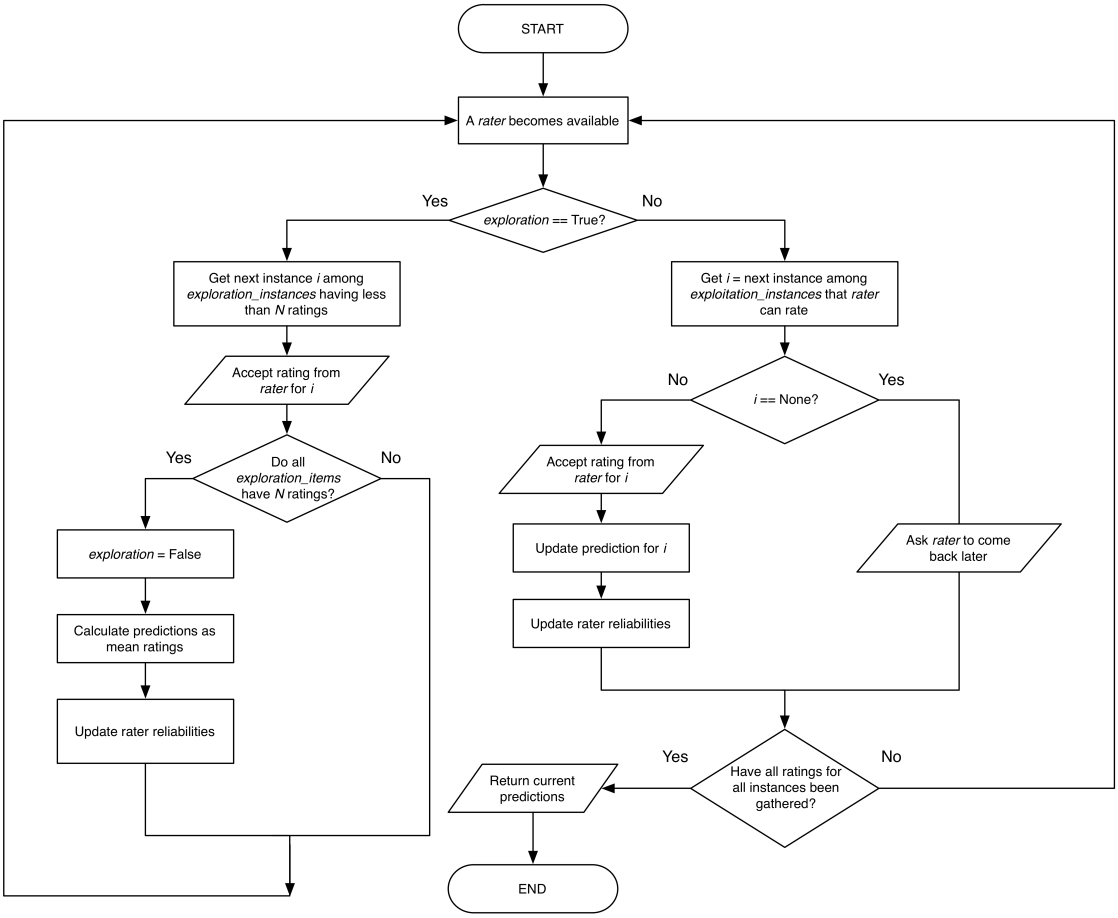


Figure 7.1: Overview of the DER^3 approach to dynamic estimation of rater reliability, when rater availability is intermittent.

of ratings needed to calculate a prediction and, therefore, start calculating rewards for any rater who rates it. If all instances have been rated by at least two raters, we take into account rater reliability. At the exploitation phase, the predictions are always calculated as average rating weighted by the reliabilities of raters who rated it to date.

Similar to the approach we developed for constant rater availability in Chapter 5, where possible, active learning is used to determine the order of presentation of instances (only the VAM datasets). The active learning algorithm is seeded with 10 instances, as previously, and then these seeds are used to calculate how informative each of the rest instances in the dataset is. Then the instances are sorted from the most informative to the least informative. The resulting sequence of instances throughout the whole rating process. If no features are available in the dataset, random order of presentation is used.

7.2 Methodology

The purpose of the experiments described in this chapter is to evaluate the DER^3 approach in the simulated rating conditions. The datasets described in Section 4.1 contain ratings for instances, where every rater rated every instance. However, details of the timings of the actual rating process are not contained in these datasets. In order to provide rater availability information we simulate rater arrivals. We assumed that every rater becomes available regularly for short periods of time and that the arrivals of raters are not correlated. We followed an approach commonly used in queuing theory where inter-arrival times are considered to be drawn from an exponential distribution. An exponential distribution has the following probability density function

$$f(x) = \lambda e^{-\lambda x}, x \geq 0, \tag{7.1}$$

where λ is often called a *rate parameter* and is inversely proportional to the mean of the distribution. In our case, the mean determines how often, on average, raters come to rate more instances. Obviously, the exact value of the mean depends on the task: in some particularly engaging and entertaining tasks it may be quite small as raters return often for the sake of enjoyment. The same can also be true for well-paid tasks. At the same time, in some tasks the instances can be uploaded in batches like in the work by Brew et al. [23], who asked raters rate news articles, once a day. Therefore, every rater arrived, on average, every 24 hours. More frequent arrivals were pointless as a new portion of articles would not be available. In our simulations the mean was equal to 8 hours, thus $\lambda = 0.125$. However, algorithms would rank exactly in the same way, independent of the value of λ. We measured time in what we call *average inter-arrival time intervals* which is the time taken divided by eight. When a rater becomes available, he can rate between 3 and 7 instances, this number is uniformly distributed. In simulated experiments rater arrivals were implemented as a queue.

In our experiments with intermittent rater availability we use DER^3 as described in Section 7.1. The approach allows to measure rater reliability in different ways, as well as to determine the border between exploration and exploitation. The following variants

were tested:

1. **DER^3/ϵ-first:** which reflects the ϵ-first multi-armed bandit algorithm, where $\epsilon \cdot 100$ percents of instances coming first in the sequence are selected as *exploration_instances*. The reliability of a rater is a mean of his rewards to date, which is the normal way how reliabilities are calculated in the ϵ-first algorithm. Following our previous experiments in Chapter 5, $\epsilon = 0.10$ is used.

2. **DER^3/KL-UCB:** which reflects the KL-UCB multi-armed bandit algorithm, where there is no distinct phase of exploration, and the DER^3 algorithm starts with exploitation. As in the original KL-UCB, rater reliability is calculated as an upper-confidence bound on rewards using Kullback-Leibler divergence.

3. **DER^3/ϵ-first*:** a feature of the DER^3 is that first two ratings for each instance are accepted from any rater even at the exploitation phase. It means that we deliberately allow some portion of noisy raters, hoping that later their impact can be reduced by weighting the ratings by reliability while calculating predictions. Another approach is to allow ratings only from reliable raters by accepting only those raters whose reliability is above median reliability of all raters (i.e. raters who are in the upper half of the rater list, sorted by reliability). The DER^3/ϵ-first* approach does exactly that. All other details are same as in DER^3/ϵ-first.

4. **DER^3/KL-UCB*:** the approach is similar to DER^3/KL-UCB approach, but it accepts ratings in the same way as DER^3/ϵ-first*.

5. **First-come-first-served (baseline):** an obvious instance-driven approach, where ratings are accepted from any raters without considering their reliability. The prediction for every instance is an average of submitted ratings and is calculated when the rating process is over. It means that N ratings submitted for an instance come from those N raters who were the first to be presented the instance to rate. This approach is very quick as it does not reject any ratings, but at the same time it also results in noisy predictions.

6. **First-come-first-served* (baseline):** as in First-come-first-served, all ratings are accepted from any raters. However, instead of just averaging the ratings to calculate predictions, the rating aggregation approach by Raykar et al. [136] was applied. We used this technique as it is a mature and widely-used technique in regression tasks.

7. **Overall-Best (baseline):** ratings are accepted only from the N raters who have the highest Raykar's reliability score when the whole dataset is considered. Overall-Best requires all N raters to rate everything, but provides highly accurate predictions.

We used three VAM datasets, as well as *BoredomVideos* and *ImageWordSimilarity*. Cost, error and time have all been used as quality measures, aggregated via MAHP. All three had the same weight equal to 0.33. The main quality metric for error is average absolute error of predictions, however, we also use average class-wise accuracy in addition.

To account for different sequences of rater arrivals, we report average errors across 50 different runs. For those rating algorithms that contained a random component, Random and ϵ-first, we ran each rating experiment on each instance sequence ten times using different random seeds and reported average errors. In the experiments using the *ImageWordSimilarity* and *BoredomVideos* datasets there was an additional random component associated with the selection of instances (as they were presented in a random order). In order to compensate for this we reported averages of 100 runs of the experiment for these two datasets (in each run instances were presented in a different order).

7.3 Results

In this section we present the results of the experiment with DER^3, highlighting its limitations, as well as discussing the magnitudes of its error.

Performance of dynamic algorithms

The results of the experiment which show the values of the aggregated MAHP performance measure are presented in Table 7.1. Tables 7.2–7.6 give separate values for cost, error and time (based on which the MAHP measure was calculated) for all datasets. In all tables the best approach in every condition is given in bold. Overall, they strongly suggest that multi-armed bandits can decrease the error of predictions at a reasonable cost and in a reasonable time. The results of First-come-first-served* approach are not included as it had significant problems with convergence[2].

All techniques behaved in the same way in our experiments. Figure 7.2 illustrates the experiment on the *VAM_Power* dataset with $N = 7$ as a representative example. Across the different approaches, time and error differed much more than cost. The Overall-Best approach was the most time-consuming as each best rater had to rate the whole dataset. DER^3/ϵ-first* and DER^3/KL-UCB* approaches used about twice as much time as their analogues DER^3/ϵ-first and DER^3/KL-UCB. Considering error, the Overall-Best approach was the best (which is not surprising), while First-come-first-served was the worst. Both DER^3/KL-UCB and DER^3/ϵ-first exhibited similar error, but they were both outperformed by DER^3/KL-UCB* and DER^3/ϵ-first*. That is to be expected as the latter approaches filter noisy ratings.

The average ranks of the approaches based on cost-error-time MAHP measure were the following (approaches are given from best to worst):

1. DER^3/ϵ-first (1.83)
2. DER^3/KL-UCB (2.10)
3. First-come-first-served (3.52)
4. DER^3/ϵ-first* (3.93)

[2]As mentioned in Section 3.3.1, static approaches can often have difficulties when ratings are sparse, i.e. every rater does not rate a big number of instances. As the EM algorithm has to estimate a lot of parameters, such sparse data can prevent its convergence. That is exactly what we have seen in our experiments with First-come-first-served* approach.

Table 7.1: The results of the experiment with intermittent availability of raters. MAHP metric values are given for each approach, the best approach is marked in bold.

Dataset	N	DER^3/ϵ-first	DER^3/KL-UCB	First-come-first-served	Overall-Best	DER^3/ϵ-first*	DER^3/KL-UCB*
BoredomVideos	3	0.14482	0.14557	**0.14347**	0.16533	0.17357	0.18172
BoredomVideos	4	0.14838	**0.14576**	0.15118	0.16513	0.17568	0.1856
BoredomVideos	5	0.15009	**0.14777**	0.15713	0.17524	0.17271	0.18489
BoredomVideos	6	0.15401	0.15321	0.16508	**0.13738**	0.17631	0.18728
ImageWordSimilarity	3	0.14976	0.14783	**0.13627**	0.1657	0.17642	0.18781
ImageWordSimilarity	4	0.15136	0.15125	**0.14343**	0.15678	0.18064	0.19384
ImageWordSimilarity	5	0.15205	**0.14881**	0.14887	0.17136	0.1752	0.19293
ImageWordSimilarity	6	**0.15374**	0.15391	0.15442	0.17449	0.17234	0.18995
VAM_Activation	3	**0.12842**	0.13015	0.13041	0.20094	0.14319	0.1535
VAM_Activation	5	**0.1373**	0.13891	0.15015	0.19179	0.15144	0.16232
VAM_Activation	7	**0.14325**	0.14468	0.16142	0.17635	0.15314	0.16573
VAM_Activation	9	**0.14847**	0.14915	0.17078	0.16071	0.15453	0.16756
VAM_Activation	11	0.15108	0.15202	0.17994	**0.1459**	0.154	0.16727
VAM_Activation	13	0.15143	0.15267	0.18628	**0.14766**	0.15078	0.16606
VAM_Activation	15	0.15264	0.15356	0.19314	0.14863	**0.14846**	0.16421
VAM_Evaluation	3	0.12587	0.12636	**0.12543**	0.20621	0.14836	0.15976
VAM_Evaluation	5	**0.13609**	0.1362	0.14197	0.198	0.15707	0.17206
VAM_Evaluation	7	**0.14054**	0.14094	0.15033	0.19001	0.16055	0.18008
VAM_Evaluation	9	0.14547	**0.14417**	0.15745	0.18131	0.16386	0.18179
VAM_Evaluation	11	0.1462	**0.14614**	0.16207	0.18091	0.164	0.18443
VAM_Evaluation	13	0.14886	**0.14819**	0.16809	0.15419	0.172	0.18629
VAM_Evaluation	15	0.15077	0.15122	0.17367	**0.12868**	0.17197	0.19158
VAM_Power	3	**0.12597**	0.12667	0.12834	0.20517	0.14638	0.15744
VAM_Power	5	**0.13676**	0.137	0.14544	0.18939	0.15702	0.1694
VAM_Power	7	0.1418	**0.14115**	0.1546	0.18602	0.16044	0.17527
VAM_Power	9	**0.14413**	0.14485	0.16238	0.17645	0.16154	0.17986
VAM_Power	11	**0.1467**	0.14695	0.16858	0.16568	0.1633	0.18171
VAM_Power	13	0.1497	0.14971	0.1756	**0.143**	0.16633	0.1834
VAM_Power	15	0.15034	0.15076	0.18102	**0.13823**	0.16575	0.18275

Table 7.2: *VAM_Activation*: the results of the experiment with intermittent availability of raters. Costs are given in the total number of ratings collected, error is average absolute error in percentage of the full ratings scale, and time is given in average inter-arrival time intervals. The best approach (based on the MAHP metric which aggregated cost, error and time) is marked in bold. Average values of cost, error and time are reported together with 95% confidence intervals.

Approaches	N = 3			N = 5			N = 7		
	Cost	Error	Time	Cost	Error	Time	Cost	Error	Time
DER^3/ϵ-first	**1403.23±0.53**	**10.98±0.13**	**10.89±0.37**	**2335.98±1.15**	**7.12±0.1**	**17.98±0.32**	**3267.15±2.32**	**5.53±0.07**	**25.29±0.33**
DER^3/KL-UCB	1403.47±0.28	11.41±0.18	10.91±0.2	2337.35±0.88	7.36±0.1	18.01±0.23	3269.77±1.63	5.72±0.09	25.18±0.33
First-come-first-served	1404.0±0.0	12.19±0.15	10.27±0.16	2340.0±0.0	9.61±0.08	17.44±0.24	3276.0±0.0	8.24±0.07	24.31±0.26
Overall-Best	1404.0±0.0	4.53±0.0	102.44±1.93	2340.0±0.0	3.38±0.0	104.11±1.8	3276.0±0.0	2.4±0.0	109.11±2.01
DER^3/ϵ-first*	1401.05±1.17	8.27±0.1	20.14±0.4	2322.25±3.65	5.19±0.06	33.39±0.48	3218.32±8.45	3.79±0.05	45.86±0.67
DER^3/KL-UCB*	1402.74±0.71	9.23±0.21	22.25±0.38	2323.48±4.82	5.61±0.1	38.1±0.66	3230.48±7.88	4.11±0.09	53.53±0.86
Approaches	N = 9			N = 11			N = 13		
	Cost	Error	Time	Cost	Error	Time	Cost	Error	Time
DER^3/ϵ-first	**4185.5±4.33**	**4.6±0.06**	**32.87±0.39**	5091.68±8.96	3.84±0.05	40.14±0.47	6000.56±9.37	3.29±0.04	46.82±0.54
DER^3/KL-UCB	4199.87±2.52	4.66±0.06	32.79±0.37	5124.94±4.37	3.92±0.05	39.81±0.38	6035.96±8.11	3.35±0.05	46.86±0.58
First-come-first-served	4212.0±0.0	7.33±0.07	31.33±0.3	5148.0±0.0	6.71±0.07	38.59±0.32	6084.0±0.0	6.24±0.06	45.61±0.37
Overall-Best	4212.0±0.0	1.75±0.0	109.15±1.98	**5148.0±0.0**	**1.24±0.0**	**110.61±2.17**	**6084.0±0.0**	**1.25±0.0**	**112.59±1.96**
DER^3/ϵ-first*	4098.15±11.7	2.95±0.03	59.1±0.73	4972.7±19.87	2.29±0.03	73.04±0.98	5674.1±37.94	1.8±0.04	89.32±1.58
DER^3/KL-UCB*	4053.98±21.85	3.26±0.06	69.09±1.02	4711.46±39.78	2.75±0.06	82.47±1.15	5075.22±59.52	2.5±0.08	96.33±1.96
Approaches	N = 15								
	Cost	Error	Time						
DER^3/ϵ-first	6873.48±15.48	2.88±0.04	53.63±0.64						
DER^3/KL-UCB	6939.98±11.8	2.89±0.03	53.9±0.51						
First-come-first-served	7020.0±0.0	5.87±0.06	52.57±0.42						
Overall-Best	7020.0±0.0	1.23±0.0	113.43±2.19						
DER^3/ϵ-first*	**5932.04±44.02**	**1.64±0.04**	**100.32±1.87**						
DER^3/KL-UCB*	5260.72±75.0	2.41±0.1	104.48±1.97						

Table 7.3: *VAM_Evaluation*: the results of the experiment with intermittent availability of raters. Costs are given in the total number of ratings collected, error is average absolute error in percentage of the full ratings scale, and time is given in average inter-arrival time intervals. The best approach (based on the MAHP metric which aggregated cost, error and time) is marked in bold. Average values of cost, error and time are reported together with 95% confidence intervals.

Approaches	N = 3			N = 5			N = 7		
	Cost	Error	Time	Cost	Error	Time	Cost	Error	Time
DER^3/ϵ-first	1403.5±0.31	9.57±0.13	10.82±0.19	**2335.28±1.83**	**6.35±0.08**	**18.04±0.34**	**3265.13±2.69**	**4.93±0.05**	**25.29±0.31**
DER^3/KL-UCB	1403.85±0.21	9.9±0.12	10.58±0.16	2338.0±0.8	6.39±0.07	17.95±0.29	3269.55±2.02	4.97±0.05	25.27±0.32
First-come-first-served	**1404.0±0.0**	**9.6±0.1**	**10.67±0.18**	2340.0±0.0	7.4±0.08	17.56±0.21	3276.0±0.0	6.2±0.06	24.58±0.25
Overall-Best	1404.0±0.0	4.43±0.0	104.29±2.15	2340.0±0.0	3.39±0.0	105.03±2.05	3276.0±0.0	2.84±0.0	109.13±1.87
DER^3/ϵ-first*	1401.3±1.25	8.42±0.17	20.27±0.36	2319.57±4.77	5.38±0.09	33.1±0.58	3227.43±7.62	4.06±0.06	46.51±0.53
DER^3/KL-UCB*	1402.34±0.67	9.64±0.24	22.14±0.54	2322.24±5.44	6.01±0.11	39.01±0.65	3218.06±9.49	4.79±0.11	55.98±0.87
Approaches	N = 9			N = 11			N = 13		
	Cost	Error	Time	Cost	Error	Time	Cost	Error	Time
DER^3/ϵ-first	4193.3±3.78	4.1±0.05	32.85±0.41	5113.12±6.03	3.48±0.04	40.1±0.43	6033.76±7.6	3.01±0.03	47.43±0.66
DER^3/KL-UCB	**4199.93±2.66**	**4.07±0.05**	**32.15±0.39**	**5122.98±5.37**	**3.5±0.05**	**39.75±0.46**	**6047.12±6.97**	**2.98±0.03**	**47.15±0.58**
First-come-first-served	4212.0±0.0	5.42±0.05	31.44±0.27	5148.0±0.0	4.91±0.04	38.58±0.34	6084.0±0.0	4.49±0.04	45.57±0.37
Overall-Best	4212.0±0.0	2.39±0.0	109.35±1.7	5148.0±0.0	2.37±0.0	111.52±1.82	6084.0±0.0	1.4±0.0	112.51±1.83
DER^3/ϵ-first*	4114.88±15.32	3.31±0.05	59.47±0.67	4940.08±31.7	2.81±0.07	72.81±1.0	5717.62±32.21	2.55±0.06	91.54±1.33
DER^3/KL-UCB*	4010.36±31.87	3.89±0.12	71.13±1.31	4617.46±59.51	3.68±0.16	84.9±1.56	4969.98±57.91	3.6±0.15	95.01±1.79
Approaches	N = 15								
	Cost	Error	Time						
DER^3/ϵ-first	6928.16±13.23	2.61±0.04	54.62±0.58						
DER^3/KL-UCB	6962.02±9.02	2.62±0.04	54.64±0.84						
First-come-first-served	7020.0±0.0	4.1±0.04	52.67±0.45						
Overall-Best	**7020.0±0.0**	**0.78±0.0**	**111.6±1.72**						
DER^3/ϵ-first*	5922.78±44.75	2.48±0.05	100.18±1.61						
DER^3/KL-UCB*	5218.7±84.53	3.77±0.36	103.75±2.27						

Table 7.4: *VAM_Power*: the results of the experiment with intermittent availability of raters. Costs are given in the total number of ratings collected, error is average absolute error in percentage of the full ratings scale, and time is given in average inter-arrival time intervals. The best approach (based on the MAHP metric which aggregated cost, error and time) is marked in bold. Average values of cost, error and time are reported together with 95% confidence intervals.

Approaches	N = 3			N = 5			N = 7		
	Cost	Error	Time	Cost	Error	Time	Cost	Error	Time
DER^3/ϵ-first	**1403.3±0.34**	**10.72±0.13**	**10.81±0.3**	**2335.85±1.37**	**7.18±0.08**	**18.35±0.33**	3266.45±1.87	5.61±0.05	25.66±0.33
DER^3/KL-UCB	1403.86±0.14	11.07±0.12	10.64±0.21	2337.83±0.84	7.24±0.09	18.28±0.31	**3270.76±1.82**	**5.61±0.07**	**25.27±0.34**
First-come-first-served	1404.0±0.0	11.75±0.1	10.43±0.14	2340.0±0.0	9.02±0.08	17.57±0.19	3276.0±0.0	7.65±0.08	24.38±0.25
Overall-Best	1404.0±0.0	4.9±0.0	103.64±1.62	2340.0±0.0	3.34±0.0	105.62±1.64	3276.0±0.0	3.04±0.0	107.47±1.64
DER^3/ϵ-first*	1400.78±1.42	9.19±0.11	19.91±0.31	2320.39±3.67	6.11±0.09	32.99±0.51	3220.97±8.86	4.57±0.07	46.44±0.66
DER^3/KL-UCB*	1402.84±0.63	10.36±0.23	21.99±0.38	2324.78±3.84	6.54±0.12	38.72±0.66	3207.84±11.56	5.09±0.12	54.73±0.8
Approaches	N = 9			N = 11			N = 13		
	Cost	Error	Time	Cost	Error	Time	Cost	Error	Time
DER^3/ϵ-first	**4196.47±3.08**	**4.57±0.05**	**32.68±0.37**	**5109.6±6.28**	**3.88±0.05**	**39.95±0.61**	6027.88±9.18	3.34±0.04	47.09±0.37
DER^3/KL-UCB	4199.03±2.87	4.68±0.06	32.38±0.47	5126.06±4.42	3.91±0.04	39.72±0.47	6057.46±4.79	3.35±0.03	46.73±0.5
First-come-first-served	4212.0±0.0	6.72±0.06	31.78±0.32	5148.0±0.0	6.12±0.07	38.31±0.34	6084.0±0.0	5.56±0.06	45.46±0.4
Overall-Best	4212.0±0.0	2.51±0.0	109.44±1.56	5148.0±0.0	1.99±0.0	111.78±2.19	**6084.0±0.0**	**1.23±0.0**	**110.28±2.0**
DER^3/ϵ-first*	4122.87±13.87	3.6±0.06	59.66±0.71	4976.9±27.97	3.03±0.09	72.67±0.99	5645.3±43.52	2.55±0.1	90.63±1.3
DER^3/KL-UCB*	4004.3±27.69	4.26±0.14	71.88±1.82	4639.18±47.61	3.87±0.12	84.37±1.31	5021.92±80.89	3.73±0.13	93.64±1.78
Approaches	N = 15								
	Cost	Error	Time						
DER^3/ϵ-first	6923.64±13.61	2.89±0.04	55.02±0.55						
DER^3/KL-UCB	6973.38±8.0	2.91±0.04	54.71±0.69						
First-come-first-served	7020.0±0.0	5.24±0.05	52.54±0.41						
Overall-Best	**7020.0±0.0**	**1.07±0.0**	**113.63±1.48**						
DER^3/ϵ-first*	5871.38±54.07	2.5±0.11	100.8±1.89						
DER^3/KL-UCB*	5142.32±94.61	3.77±0.15	102.6±2.16						

Table 7.5: *BoredomVideos*: the results of the experiment with intermittent availability of raters. Costs are given in the total number of ratings collected, error is average absolute error in percentage of the full ratings scale, and time is given in average inter-arrival time intervals. The best approach (based on the MAHP metric which aggregated cost, error and time) is marked in bold. Average values of cost, error and time are reported together with 95% confidence intervals.

Approaches	N = 3			N = 4		
	Cost	Error	Time	Cost	Error	Time
DER^3/ϵ-first	134.53±0.35	10.26±0.43	2.96±0.17	178.52±0.67	7.59±0.28	4.44±0.22
DER^3/KL-UCB	134.49±0.57	11.47±0.46	2.69±0.11	**179.8±0.14**	**8.32±0.29**	**3.81±0.17**
First-come-first-served	**135.0±0.0**	**11.49±0.41**	**2.56±0.11**	180.0±0.0	9.88±0.35	3.58±0.1
Overall-Best	135.0±0.0	3.99±0.0	11.33±0.52	180.0±0.0	3.63±0.0	12.73±0.52
DER^3/ϵ-first*	132.36±0.92	9.95±0.43	5.37±0.33	168.62±2.5	7.72±0.52	7.71±0.42
DER^3/KL-UCB*	132.48±1.36	11.51±0.47	5.33±0.25	172.45±2.38	8.57±0.38	8.02±0.39
Approaches	N = 5			N = 6		
	Cost	Error	Time	Cost	Error	Time
DER^3/ϵ-first	221.44±1.2	6.04±0.23	5.87±0.25	260.02±2.33	4.98±0.2	7.43±0.43
DER^3/KL-UCB	**224.08±0.52**	**6.49±0.22**	**5.15±0.23**	266.4±1.63	5.29±0.2	6.72±0.25
First-come-first-served	225.0±0.0	8.89±0.31	4.51±0.13	270.0±0.0	8.04±0.25	5.47±0.14
Overall-Best	225.0±0.0	4.24±0.0	13.16±0.53	**270.0±0.0**	**1.82±0.0**	**13.85±0.45**
DER^3/ϵ-first*	191.69±4.1	6.79±0.63	9.23±0.46	206.67±5.83	6.53±0.7	10.74±0.62
DER^3/KL-UCB*	209.02±3.49	6.71±0.33	10.53±0.5	231.82±4.31	6.03±0.39	12.45±0.54

Table 7.6: *ImageWordSimilarity*: the results of the experiment with intermittent availability of raters. Costs are given in the total number of ratings collected, error is average absolute error in percentage of the full ratings scale, and time is given in average inter-arrival time intervals. The best approach (based on the MAHP metric which aggregated cost, error and time) is marked in bold. Average values of cost, error and time are reported together with 95% confidence intervals.

Approaches	N = 3			N = 4		
	Cost	Error	Time	Cost	Error	Time
DER^3/ϵ-first	247.95±0.52	10.14±0.28	5.59±0.21	328.91±0.92	7.36±0.2	7.91±0.25
DER^3/KL-UCB	248.81±0.21	10.17±0.28	5.34±0.17	330.37±0.62	7.51±0.25	7.7±0.28
First-come-first-served	**249.0±0.0**	**8.25±0.16**	**5.14±0.15**	**332.0±0.0**	**7.09±0.15**	**6.91±0.19**
Overall-Best	249.0±0.0	3.8±0.0	20.18±0.7	332.0±0.0	2.98±0.0	21.53±0.65
DER^3/ϵ-first*	242.36±1.85	9.35±0.44	10.19±0.41	308.09±4.36	7.49±0.48	14.18±0.6
DER^3/KL-UCB*	245.26±1.41	9.99±0.33	11.39±0.41	316.15±3.73	7.96±0.35	16.1±0.49
Approaches	N = 5			N = 6		
	Cost	Error	Time	Cost	Error	Time
DER^3/ϵ-first	407.59±1.58	5.81±0.21	9.9±0.35	**482.33±3.0**	**4.52±0.13**	**12.78±0.39**
DER^3/KL-UCB	**413.1±0.7**	**5.76±0.16**	**9.23±0.26**	491.85±1.69	4.64±0.12	12.25±0.35
First-come-first-served	415.0±0.0	5.96±0.12	8.89±0.2	498.0±0.0	5.34±0.1	10.62±0.2
Overall-Best	415.0±0.0	3.65±0.0	22.62±0.75	498.0±0.0	3.58±0.0	22.94±0.64
DER^3/ϵ-first*	350.36±6.06	5.82±0.3	17.97±0.69	356.96±7.56	5.92±0.43	18.64±0.77
DER^3/KL-UCB*	376.36±6.82	6.53±0.29	19.97±0.7	402.78±9.4	6.06±0.34	21.67±0.83

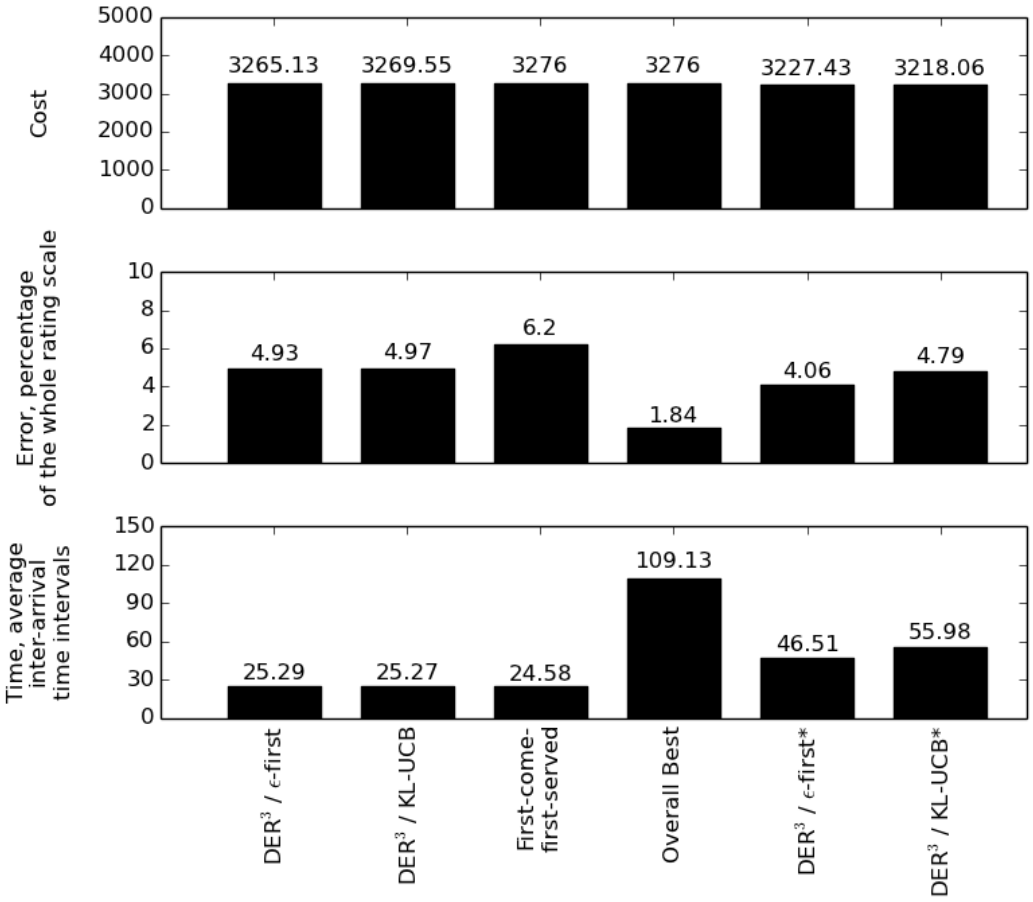


Figure 7.2: Comparative performance of rater reliability estimation algorithms at the end of the rating process (*VAM_Evaluation*, $N = 7$).

5. Overall-Best (4.07)

6. DER^3/KL-UCB* (5.55)

DER^3/KL-UCB and DER^3/ϵ-first were the best approaches as they offer low error predictions for a reasonable time. Overall-Best, DER^3/ϵ-first* and DER^3/KL-UCB* required a lot of time to gather ratings and ranked the worst, despite leading to a lower error compared to other DER^3 algorithms.

According to the results of the Friedman test, there is a statistically significant difference between the approaches (p-value $= 5.08 \times 10^{-11}$). The post-hoc Bergmann-Hommel test was able to find two distinct groups: one consisting of DER^3/ϵ-first and DER^3/KL-UCB, and another containing the rest of algorithms ($\alpha = 0.05$). This indicates that both DER^3/KL-UCB and DER^3/ϵ-first are significantly better than other approaches used in

our experiments. These two approaches in many cases gathered less than N ratings per instance and still were able to result in ratings with a low error. This indicates that using less than N ratings can be beneficial for the scenario of rater intermittent availability and presents interesting opportunities for future work.

Limitations of DER^3

Although in the majority of our experiments DER^3/ϵ-first or DER^3/KL-UCB were the best approaches, they showed poor performance when very high and very low values of N were used.

When N is high, the collection of ratings takes a long time no matter which approach is used. Thus, the advantage in time that DER^3 had over the Overall-Best and DER^3* approaches was reduced. At the same time, the Overall-Best and DER^3* approaches achieved the highest value of the MAHP metric as they still produced much lower error than DER^3/ϵ-first or DER^3/KL-UCB. So, DER^3* approaches have some potential only when the N is large (about 50% or more of the overall number of raters in our experiments).

For experiments when N is low, the poor performance of DER^3/KL-UCB and DER^3/ϵ-first is explained by the fact that these approaches accepted the first two ratings for each instance from any rater. Thus, when N was equal to three or four, 66% or 50% respectively of all ratings were collected without any consideration of rater reliability. This resulted in predictions of poor quality, which made it difficult to estimate the rater reliability precisely. We use the *ImageWordSimilarity* dataset to illustrate this problem.

Figure 7.3 shows how the estimates of rater reliability changed as raters rated instances from the *ImageWordSimilarity* dataset when DER^3 was used for different values of N. The vertical axis represents reliability (the higher the better), the horizontal axis shows the time, and each line represents a single rater. The reliabilities of raters change as new ratings arrive and predictions are updated. Ideally, reliable raters should be detected as early as possible. However, DER^3/ϵ-first ranked the four most reliable raters (denoted by black lines) lower than other, noisier raters (grey lines) (Figure 7.3a). Figure 7.3c illustrates a similar problem with DER^3/KL-UCB used in the same setting. However,

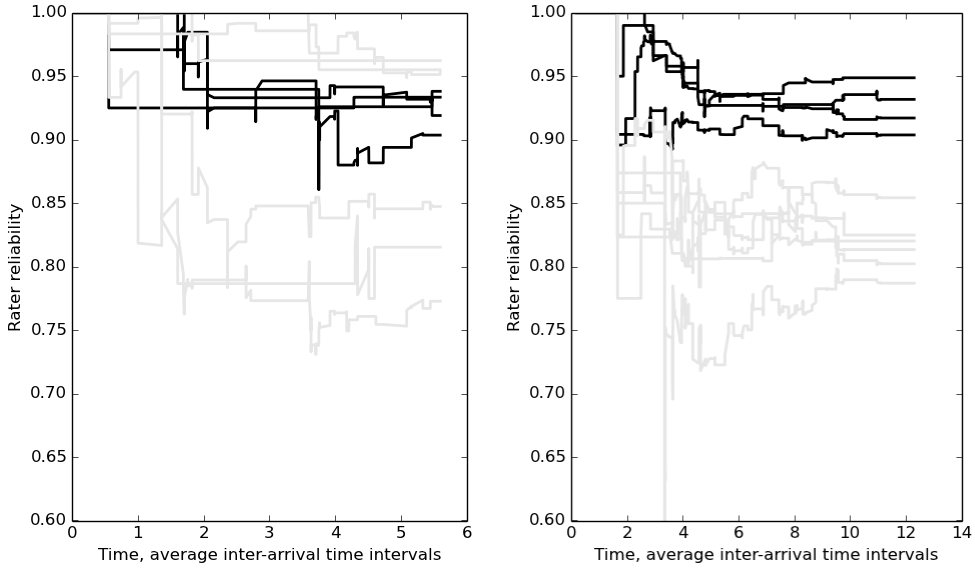
when N was higher, additional ratings compensated for the poor quality of the first two ratings. Figures 7.3b and 7.3d illustrate how rater reliability changed when the experiment was conducted on *ImageWordSimilarity* dataset with $N = 6$. Both DER^3/KL-UCB and DER^3/ϵ-first were able to rank the top four raters (black lines) close to the top of the rater population. These results suggest that the DER^3 approach should not be used when the N is small (equal to three in our experiments).

Discussion of errors

In many experiments the difference in error between the best and the worst approach was just about 3–7% of the whole rating scale. At first glance it might look like a very minor improvement. However, the average absolute error does not take into account the rating imbalance that exists in the datasets. To take this into account, we looked at these errors from another angle, considering them in terms of a classification problem. The rating scale in the VAM datasets was considered as an ordinal classification problem. For instance, in the VAM datasets the ratings come from the [-1, -0.5, 0, 0.5, 1] set. The rating predictions were rounded to the nearest point on the rating scale, for instance, in VAM datasets 0.48 would be rounded to 0.5. The same was done with the gold standard ratings. When the average class-wise accuracy was used as a performance measure, the ranking of the different approaches to rater selection did not change; however, the magnitude of the differences between the performance of the algorithms became more evident.

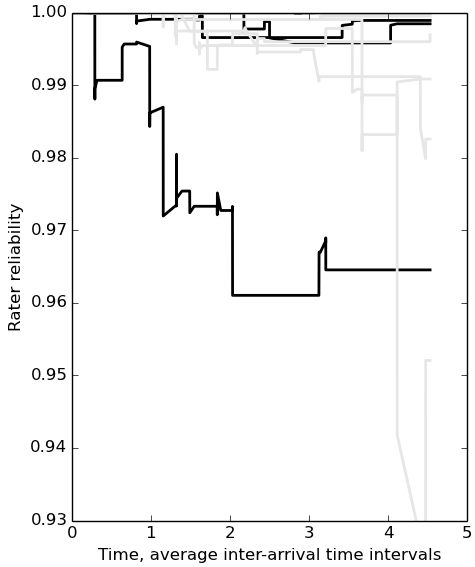
We discuss a run of DER^3/ϵ-first on the *VAM-Activation* dataset with $N = 9$ as an illustrative example (we witnessed a very similar situation across all datasets and all values of N). This run resulted in an error of 4.28%, while the First-come-first-served baseline achieved an error of 7.17%. The confusion matrices from the corresponding classification problems are displayed in Tables 7.7 and 7.8.

These tables show that the difference between the performance of these approaches was 16 percentage points in terms of average class-wise classification accuracy. While the accuracy on the majority class was close in both approaches (74% vs. 80%), they differed on the other classes. The aggregated answers of the First-come-first-served raters

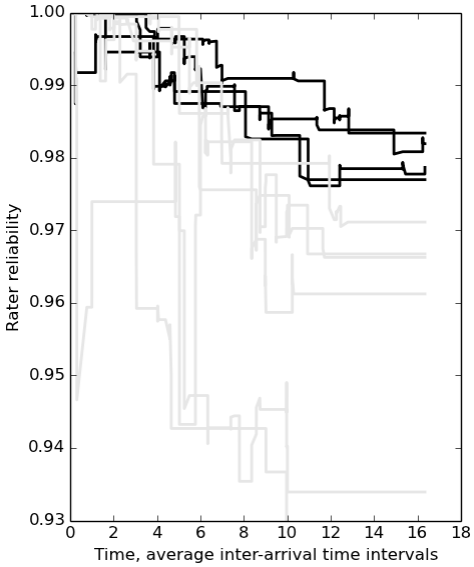


(a) DER^3/ϵ-first, $N = 3$

(b) DER^3/ϵ-first, $N = 6$



(c) DER^3/KL-UCB, $N = 3$



(d) DER^3/KL-UCB, $N = 6$

Figure 7.3: Changes of rater reliability in a single run of the simulated rating experiment on *ImageWordSimilarity* dataset.

correctly identified only 36% of very active instances compared to 79% by raters selected using ϵ-first.

As very active instances constitute only about 6% of the whole dataset, these errors became much less noticeable when average absolute error was used as a performance mea-

sure. At the same time predictions of a good quality should be precise enough for instances across the full rating scale, not only around the majority of ratings. It is especially important in the area of emotion recognition from speech where getting "minority class" recordings with extreme levels of activation (e.g. when a person is very happy or very angry) is difficult. As a result, a training set for an emotion recogniser can be biased, as is the case with VAM datasets. Many real-life applications such as detection of anger in calls to call centres demand a reliable detection of such extreme events, which is impossible without having a correctly labelled dataset.

Table 7.7: Confusion matrix for a single run on *VAM_Activation* ($N = 9$), DER^3/ϵ-first.

		Predicted						
		-1	**-0.5**	**0**	**0.5**	**1**	**Total**	**Accuracy**
Actual	**-1**	0	0	0	0	0	0	–
	-0.5	0	43	15	0	0	58	74%
	0	0	28	195	21	0	244	80%
	0.5	0	0	17	111	10	138	80%
	1	0	0	0	6	22	28	79%
								Average: 78%

7.4 Conclusions

In this chapter we presented and evaluated DER^3, a novel approach to dynamic estimation of rater reliability based on multi-armed bandits. The evaluation was performed under the realistic assumption of intermittent availability, i.e. when raters can start and finish rating at intermittent and variable times.

DER^3 waits for a rater to become available and then decides which instances to give him to rate. The general idea is that a rater is asked to rate instances that have previously been rated by raters who are less reliable than the available rater. Both DER^3/ϵ-first and DER^3/KL-UCB (the methods that are based on multi-armed bandit approaches) showed significantly better performance than the baseline approach of accepting ratings from any

Table 7.8: Confusion matrix for a single run on *VAM_Activation* ($N = 9$), First-come-first-served.

		Predicted						
		-1	**-0.5**	**0**	**0.5**	**1**	**Total**	**Accuracy**
Actual	**-1**	0	0	0	0	0	0	–
	-0.5	1	41	16	0	0	58	71%
	0	0	53	181	10	0	244	74%
	0.5	0	0	37	95	6	138	69%
	1	0	0	0	18	10	28	36%
								Average: 62%

rater who was available. The performance measure used combined the cost (the number of ratings collected), the accuracy of the rating predictions and the time needed to collect the ratings. DER^3 proved to be fast, cost-effective and accurate. We found that DER^3 works especially well in typical crowdsourcing conditions when 5–10 raters rate each instance. We also tried more noise-intolerant variations of DER^3 (DER^3/ϵ-first* and DER^3/KL-UCB* approaches) which was as accurate, but required more time to get the necessary number of ratings. We have some evidence that shows that these more noise-tolerant approaches are good when larger numbers of raters can be used.

We used an Overall-Best baseline, using the subset of the best raters based on their performance on the whole dataset. In real-life conditions this would correspond to the situation when we know the best raters in advance and ask them to rate every instance. The accuracy of the predictions based on such a subset of raters usually was higher than that achieved by the multi-armed bandit techniques, which suggests that there is room for improving DER^3.

In the next chapter we present the results and discussion of using DER^3 on the Amazon Mechanical Turk platform to see whether this approaches works in real-life crowdsourcing conditions.

Chapter 8

Real-life Evaluation of Dynamic Estimation of Rater Reliability

Our previous experiments (Chapter 7) validated the DER^3 approach using simulations of the scenario of intermittent rater availability. The goal of the experiments in this chapter is to conduct additional validation using a real-life crowdsourcing platform, Amazon Mechanical Turk (AMT). We performed an experiment that compared the accuracy of ratings for a set of emotional speech recordings collected using a standard first-come-first-served AMT approach with a set collected using DER^3 to determine the effectiveness of DER^3. We placed 160 speech recordings on AMT, and asked workers to rate them on the scales of Activation and Evaluation. The typical AMT approach was applied: every rater was allowed to rate as many instances as desired, and reliability was never tracked or estimated. We gathered seven ratings for each instance on each of the two scales. We then repeated the rating process, but this time, we tracked rater reliability using the DER^3 approach. We estimated the accuracy of predictions given by both approaches, by comparing the results to the gold standard that was also gathered on AMT by using output from 30 additional workers.

This chapter is structured as follows. Section 8.1 covers the methodology of the experiments. The results are covered in Section 8.2, while Section 8.3 concludes the chapter.

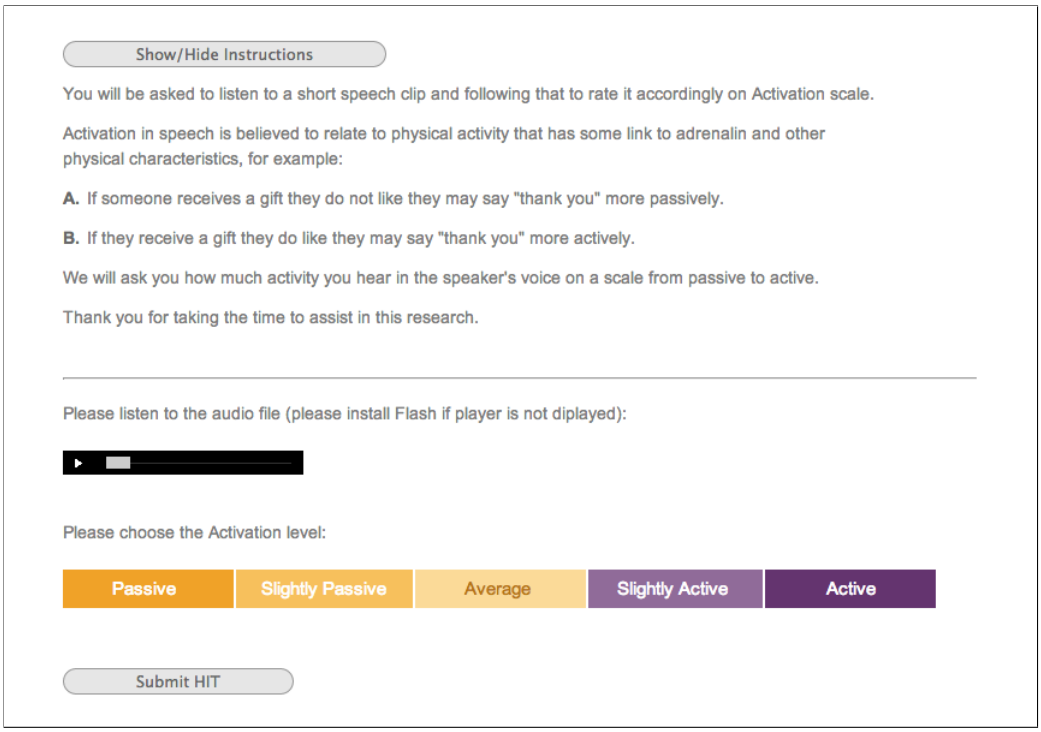
8.1 Methodology

For our experiments we used a corpus of emotional speech recordings collected by Vaughan [186]. It is an elicited emotional speech corpus, containing recordings of native English speakers from Ireland playing a game. In order to get authentic depictions of emotion, a mood-induction procedure was applied in the form of a shipwreck game: participants were the only survivors of a shipwreck, and had 10 minutes to rank 15 items (such as flares, a radio or water) from the most useful to the most useless. Each participant was placed into a separate soundproof booth to ensure good-quality recording. Participants communicated with each other via headsets, and all conversation were recorded. Sixteen participants took part in the experiment in pairs, and the total length of audio recorded was 150 minutes. From these recordings 160 short clips were extracted to be used as a dataset. We used all 160 instances in our AMT experiments.

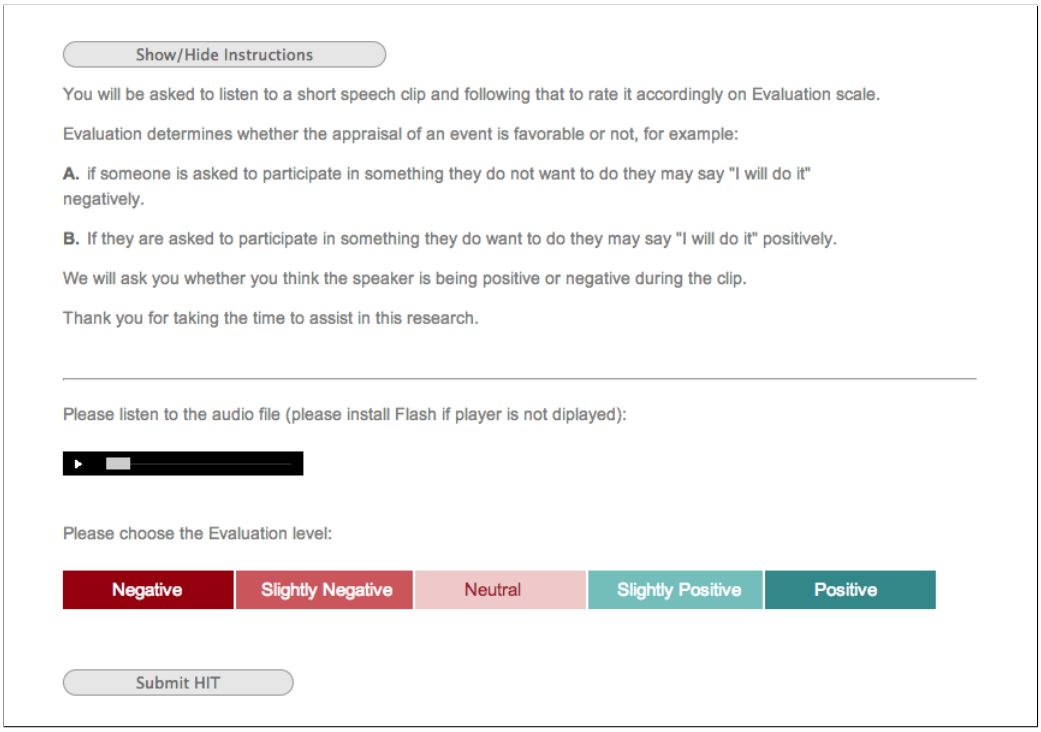
In order to get gold standard ratings for these recordings, we developed a specialised rating interface for this task. This contained a player and buttons with possible ratings, as well as rating instructions. The interface is shown in Figure 8.1. The design of the interface is somewhat similar to the rating tool used in rating the VAM corpora (Section 2.2.3), because the set of ratings was similar to that used in VAM: we also have five levels on both the Activation and Evaluation dimensions. The instructions presented to raters contained a brief explanation of both dimensions, with examples. A detailed overview of how the interface was developed and tested can be found in Appendix B.

In order to get gold standard ratings, we placed all 160 instances on Amazon Mechanical Turk, where raters submitted their work using the interface described above. Usually, when reliable predictions are needed, a large number of ratings is gathered for each instance. This number rarely exceeds 20, but we gathered 30 ratings for each recording, to be confident in the quality of the predictions. All the ratings were mapped to the numerical [0, 1, 2, 3, 4] scale as given in Table 8.1. The gold standard for each instance was calculated as an average rating[1]. The distribution of the gold standard, showed in

[1]We also tried to apply the rating aggregation approach by Raykar et al. [136], but it gave very similar results.



(a) Activation

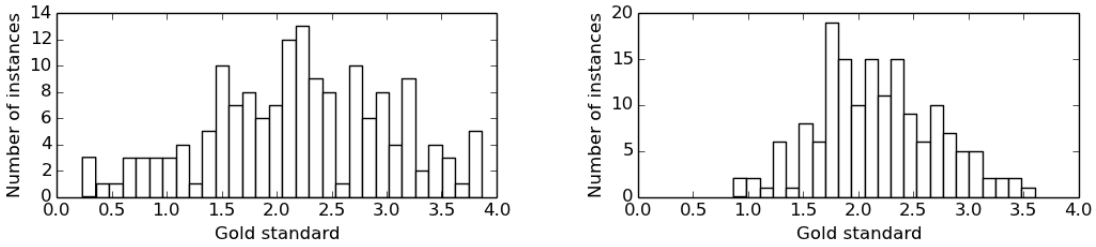


(b) Evaluation

Figure 8.1: AMT interface for rating emotional speech.

Table 8.1: Mapping of Activation and Evaluation classes to the numerical scale.

Numerical scale	Activation class	Evaluation class
0	Passive	Negative
1	Slightly Passive	Slightly Negative
2	Average	Neutral
3	Slightly Active	Slightly Positive
4	Active	Positive



(a) Activation

(b) Evaluation

Figure 8.2: Distribution of gold standard ratings. Each rating was an average of 30 ratings submitted by AMT workers. The discrete classes given to raters map to the [0, 1, 2, 3, 4] scale, where 0 represents Negative/Passive and 4 represents Positive/Active classes.

Figure 8.2, is typical for the domain of natural emotional speech: instances are centered around the middle of the scale, representing neutral speech. However, as the subjects in the recordings were under time pressure, they were more active than passive. Because a game was used to induce emotions, participants did not tend to get very negative.

We implemented the DER^3 approach as outlined in the previous chapter. The details of implementing it to work with AMT are given in Appendix C. In order to evaluate the performance of the DER^3 approach, we compared it to a standard First-come-first-served AMT approach, where all raters are allowed to rate as many instances as they want. The main idea of this experiment is to gather two separate sets of ratings using both approaches, and comparing them to the gold standard.

We collected seven ratings for each instance on each dimension, as this number is close to the number of ratings typically gathered in crowdsourcing tasks. All instances were placed online, and any rater could rate as many of them as desired. This is a fairly typical

Table 8.2: Average absolute errors of predictions (expressed as the percentage of the full rating scale) of First-come-first-served and DER^3 approaches.

	Activation	Evaluation
First-come-first-served	13.78%	9.64%
DER^3	9.56%	9.56%

First-come-first-served AMT scenario. A prediction for each instance was an average of ratings submitted by raters[2].

The same process was then repeated, but this time rater reliability was tracked dynamically using DER^3/KL-UCB as a representative example (in this chapter we refer to it as DER^3 for simplicity). In it the prediction is an average rating weighted by reliabilities of raters who rated it, as described in Section 7.1.

To eliminate any learning effects in the experiment, we used two separate sets of AMT raters for the first-come-first-served and DER^3 experiments.

To determine the sequence of instances for DER^3, we used active learning, as described in Section 7. The active learning process was seeded with the 10 most informative instances. Thus, during the rating process, ratings were collected for 150 instances out of 160.

In order to evaluate the performance of the first-come-first-served and DER^3 approaches, we compared the predictions made with the gold standard, using both average absolute error and average class-wise classification accuracy. Classes were mapped to the [0, 1, 2, 3, 4] numerical scale. We did not consider cost, as both first-come-first-served and DER^3 gathered exactly the same number of ratings. We paid \$0.01 for every rating gathered. The total budget of the experiment was \$132.

8.2 Results

The errors of prediction for DER^3 and first-come-first-served are given in Table 8.2. DER^3 achieved a lower error on Activation, while its performance on Evaluation was similar to that of first-come-first-served.

The average error might not give a complete understanding of the performance of both approaches as the gold standard for both dimensions was concentrated in the middle of the rating scale. This means that if a rater can not pick any emotion from speech and rates all instances as Average/Neutral, he is often correct. In order to investigate these errors more thoroughly, we also calculated the average class-wise accuracy achieved by each approach. When no estimation of rater reliability was carried out, the average class-wise classification accuracy for Activation was only 30%. Accuracy for Evaluation was slightly higher, but still quite low: 44%. When DER^3 was used, the accuracies rose to 69% and 72% for Activation and Evaluation respectively. More detailed results are presented as confusion matrices in Figure 8.3. The accuracy for the majority class is never above 67% and is the same for both the DER^3 and first-come-first-served approaches. This is indicative of the difficulty inherent in recognising emotion from natural speech. The difference between approaches manifests itself in other classes, where the number of instances is comparatively low. For instance, predictions made by first-come-first-served raters were incorrect for all of the passive instances (class 0), while predictions calculated by DER^3/KL-UCB were correct for 80% of these instances. A similar situation can also be seen for the other non-majority classes. Thus, our results suggest that using DER^3 can improve the accuracy of the ratings collected via crowdsourcing when a platform similar to AMT is used.

In DER^3 49 different raters participated in rating instances on the Activation scale. Evaluation tasks turned out to be somewhat less popular: only 29 raters performed them. This may be due to several factors. For instance, the word "evaluation" might have been

[2]For regression tasks, the rating aggregation approach by Raykar et al. [136] usually gives more precise results than just averaging, but in our experiments in Section 7, we discovered that it can have problems with converging in scenarios where ratings are sparse. As ratings gathered on AMT are often sparse, we decided not to apply the algorithm by Raykar et al. [136], and used averaging instead.

associated with a particular kind of task that raters preferred to avoid. Nevertheless, the overall rating process looked very similar for both emotional dimensions. Figures 8.4 and 8.5 show how ratings were collected for Activation and Evaluation respectively. Every circle represents an event when a rater became available, an instance selected, and a rating submitted. A cross represents an event when a rater was willing to rate, but there were no suitable instances available. Time when a particular event occurred (GMT+0 time zone) is given on the horizontal axis, while the vertical axis represents a rater's identification number.

There was no distinct pattern in how raters became available. Some raters returned to tasks from time to time (for example, raters #49, #17 and #23 on the Activation dataset), while others rated a sequence of instances and then never came back. Many of raters rated a relatively large number of instances (raters #3 or #41 on the Activation dataset were amongst the most prolific). Usually, some time was required for DER^3 to realise that a certain rater was not accurate: for instance, rater #38 was allowed to submit quite a few ratings before his first rejection. It is interesting to note that if a rater got rejected, he usually never returned. As should be expected, only raters with low reliability were not allowed to rate at some stage of the rating process. Figure 8.6a shows the reliabilities of all raters at the end of the process, in descending order. Each reliability is calculated as in KL-UCB multi-armed bandit algorithm, i.e. as an upper-confidence bound of rewards using Kullback-Leibler divergence. Black bars represent raters who were always were presented with an instance to rate, upon becoming available. White bars denote raters who were rejected at least once. Figure 8.6b shows the same for the Evaluation dimension.

The simulated experiments In Chapter 7 relied on two assumptions about rater availability:

1. Every rater becomes available from time to time.

2. When a rater becomes available, he rates a small number of instances (on average, five).

In order to check how well these assumptions work on AMT, we had to determine

when a certain rater becomes available and unavailable. Unfortunately, AMT does not allow such events to be tracked; however, this information can be inferred from the rating timestamps. If there is a big distance in time between two ratings, we can conclude that the rater became unavailable after the first rating and then returned. We can also count how many instances the rater rated in a batch before becoming unavailable, as well as the distance between such batches. We considered that if the distance between two ratings is smaller than 60 seconds, they belong to the same batch. Therefore, if a rater had not submitted a rating in 60 seconds, we assumed that he became unavailable.

Most raters in our experiments rated only one batch of instances: that was the case with 79% of raters on Evaluation and 59% on Activation. This can be explained by the large variety of tasks that is offered on AMT; a rater might prefer to switch to a different task when the current task gets boring or repetitive. On average, each rater did 2.76 batches on Activation and 1.89 batches on Evaluation. A typical distance between two batches was less than half an hour (Figure 8.7), i.e. if a rater returned, he did so shortly after rating the previous batch. Figure 8.8 shows the distribution of a batch size: there were very few attempts to rate more than 20–30 instances in one go. In fact, most batches consisted of 10 or fewer instances. This means that while assumption #2 was true in many cases, assumption #1 was usually violated. However, despite this, DER^3 was able to achieve better results than the baseline.

8.3 Conclusions

In this chapter we performed additional evaluation of DER^3 using the AMT platform. In the experiments, the DER^3 approach proved better than a traditional AMT approach (First-come-first-served baseline), where any rater can rate as many instances as desired, and no rater reliability is considered during the rating process. These results suggest that DER^3 can be useful in real-life crowdsourcing scenarios.

In our earlier experiments in Chapter 7, we assumed that every rater becomes available regularly, and is able to rate a small number of instances every time. In the experiments

in this chapter, raters generally did not return, or returned only rarely. Nevertheless, even with this assumption being violated, DER^3 managed to get predictions of a higher accuracy than these calculated by first-come-first-served.

Figure 8.3: Confusion matrix for AMT experiments. Classes are represented as numbers from 0 (Negative/Passive) to 4 (Positive/Active).

(a) Activation, First-come-first-served

		Predicted						
		0	**1**	**2**	**3**	**4**	**Total**	**Accuracy**
Actual	**0**	0	2	3	0	0	5	0%
	1	0	13	12	2	0	27	48%
	2	0	10	39	14	0	63	62%
	3	0	0	27	14	0	41	34%
	4	0	0	6	7	1	14	7%

Average: 30%

(b) Activation, DER^3/KL-UCB

		Predicted						
		0	**1**	**2**	**3**	**4**	**Total**	**Accuracy**
Actual	**0**	4	1	0	0	0	5	80%
	1	2	22	3	0	0	27	81%
	2	0	8	38	17	0	63	60%
	3	0	1	9	27	4	41	66%
	4	0	0	0	6	8	14	57%

Average: 69%

(c) Evaluation, First-come-first-served

		Predicted						
		0	**1**	**2**	**3**	**4**	**Total**	**Accuracy**
Actual	**0**	0	0	0	0	0	0	–
	1	1	10	7	0	0	18	56%
	2	1	12	66	24	0	103	64%
	3	0	2	5	15	6	28	54%
	4	0	0	0	1	0	1	0%

Average: 44%

(d) Evaluation, DER^3/KL-UCB

		Predicted						
		0	**1**	**2**	**3**	**4**	**Total**	**Accuracy**
Actual	**0**	0	0	0	0	0	0	–
	1	1	11	6	0	0	18	61%
	2	0	13	69	21	0	103	67%
	3	0	0	8	17	3	28	61%
	4	0	0	0	0	1	1	100%

Average: 72%

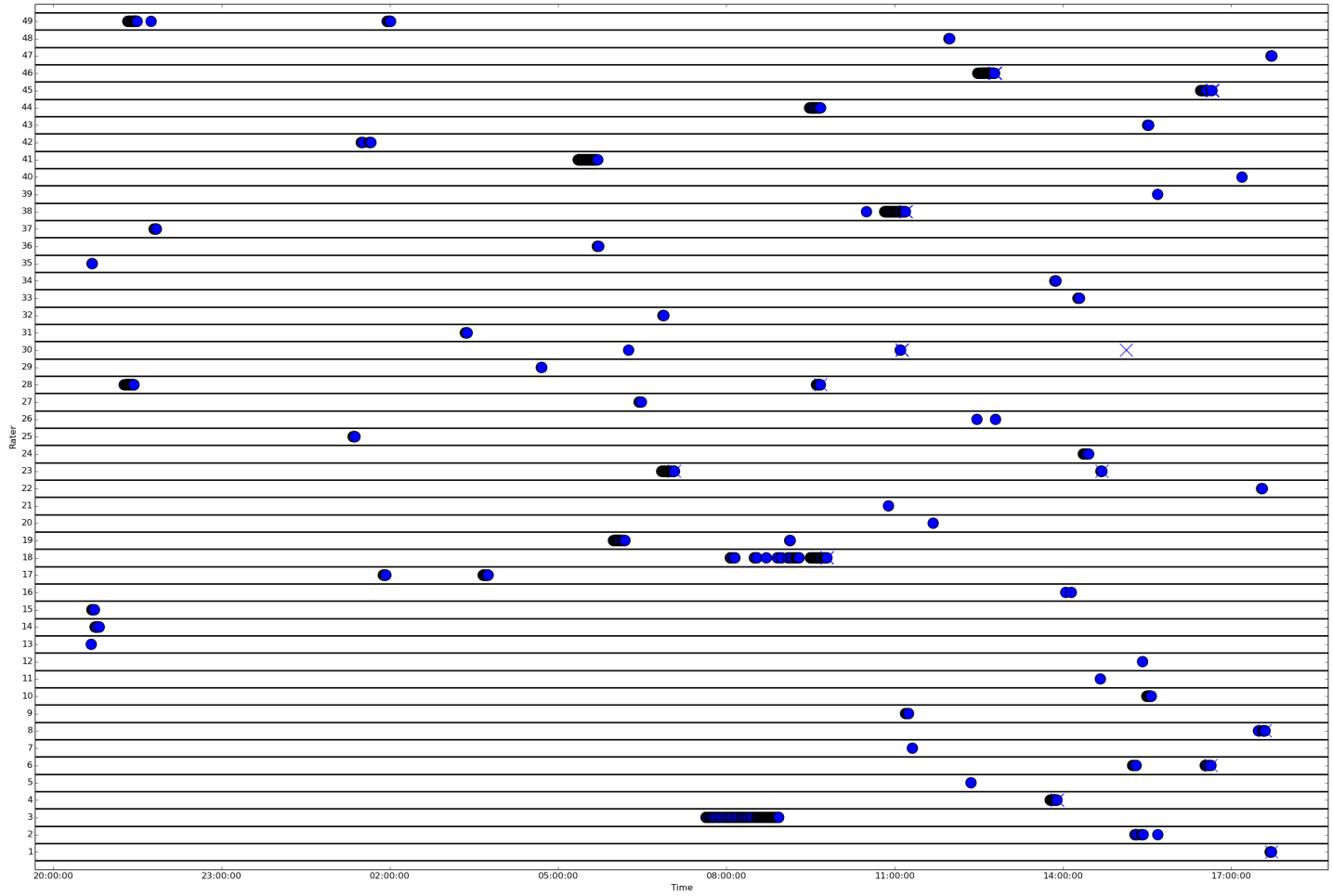


Figure 8.4: The rating process for Activation dimension when DER^3 was used. Every circle represents an event when a rater became available, an instance was selected, and a rating submitted. A cross represents an event when a rater was willing to rate, but no suitable instances were available. Time when a particular event occurred (GMT+0 time zone) is given on X axis, while Y axis represents a number of rater.

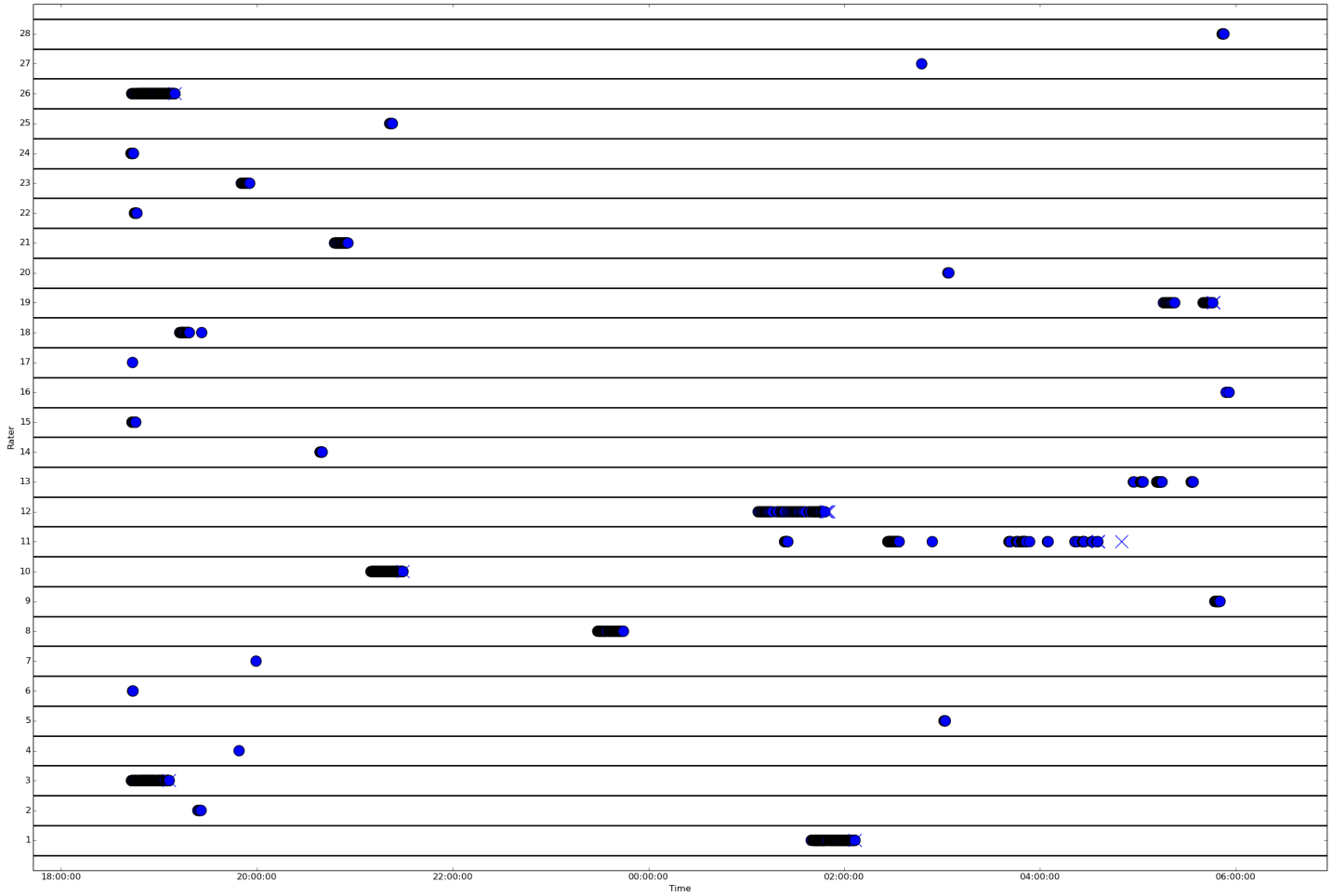
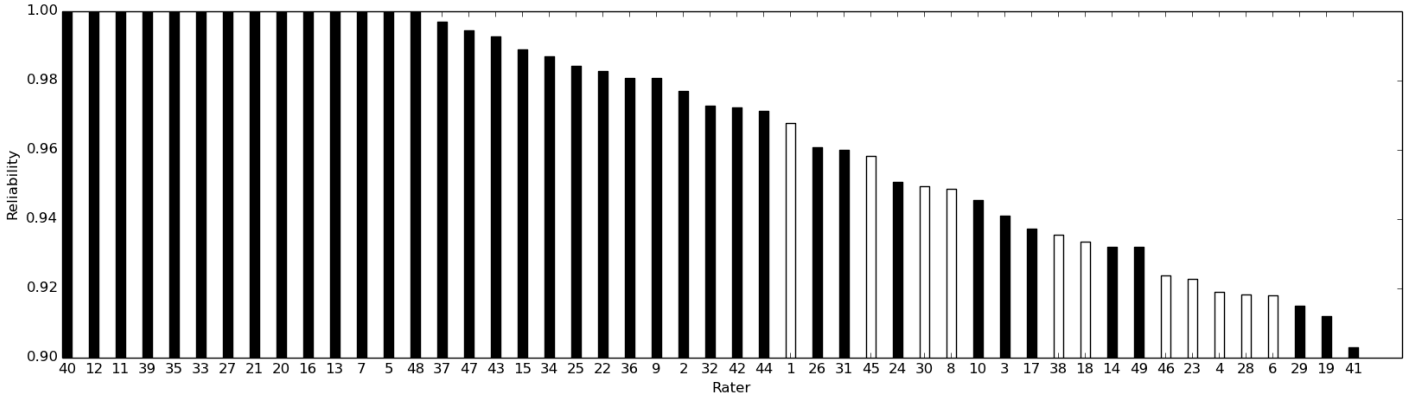
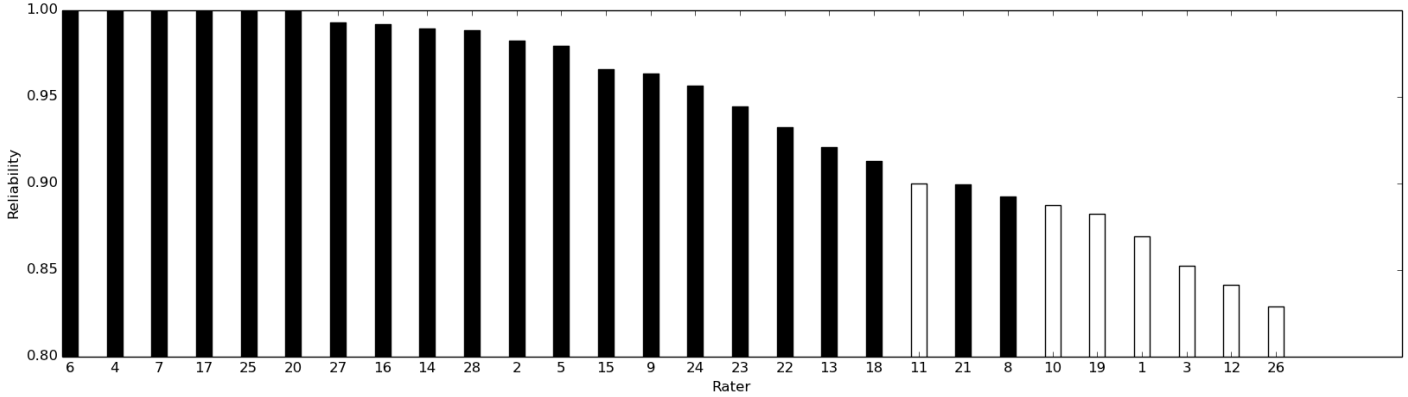


Figure 8.5: The rating process for Evaluation dimension when DER^3 was used. Every circle represents an event when a rater became available, an instance selected, and a rating submitted. A cross represents an event when a rater was willing to rate, but no suitable instances were available. Time when a particular event occurred (GMT+0 time zone) is given on X axis, while Y axis represents a number of rater.

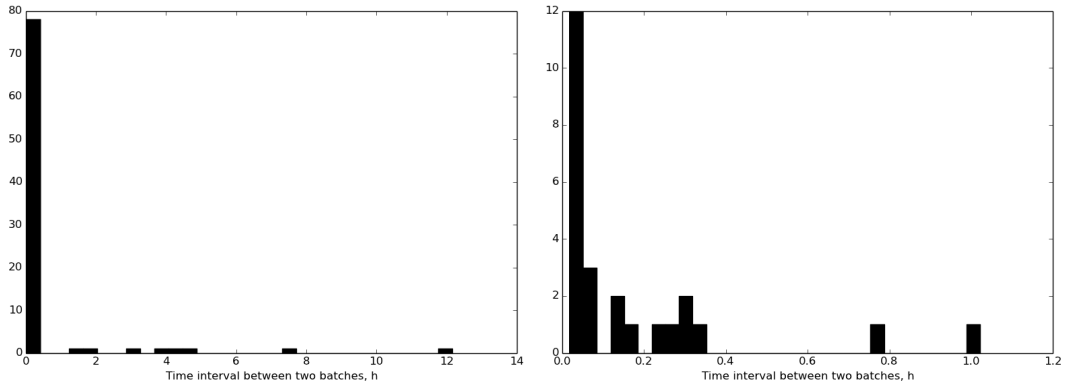


(a) Activation



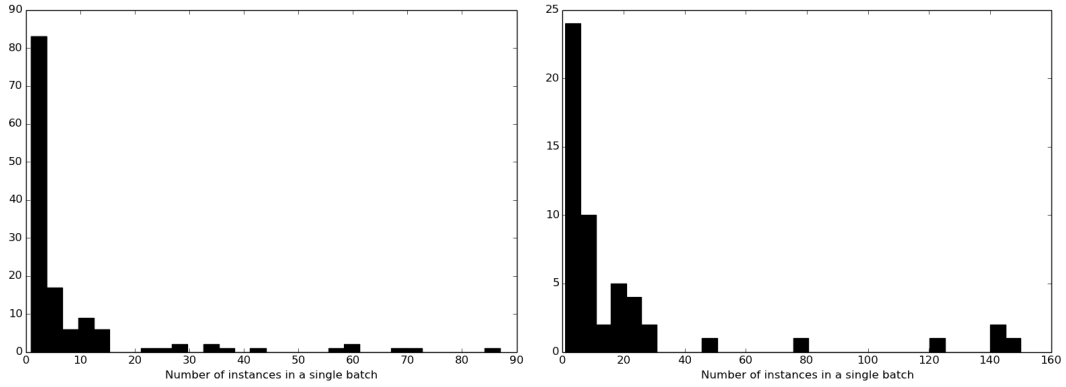
(b) Evaluation

Figure 8.6: Final KL-UCB reliabilities of all raters who participated in experiments in descending order. Black bars represent raters who were always presented with an instance to rate, while white bars denote those whose rating were rejected at least once. There is no correspondence between rater numbers in the two figures, i.e. rater #1 on Activation is not necessarily the same person as rater #1 on Evaluation.



(a) Activation (b) Evaluation

Figure 8.7: Distributions of distances in time between two batches from a same rater (a batch is a sequence of instances rated by a rater in one session).



(a) Activation (b) Evaluation

Figure 8.8: Distributions of batch sizes (a batch is a sequence of instances rated by a rater in one session).

Chapter 9

Conclusions

Crowdsourcing has been successfully applied to solving problems from different areas, but the existence of unreliable and noisy workers still poses a significant challenge. This is also true for supervised machine learning, where crowdsourcing is often used or even required to collect ratings for training instances. For crowdsourcing tasks where the price for a single rating is high, it makes more sense to estimate rater reliability dynamically, rather than to get all possible ratings, and do it at the end of the rating process. In this thesis, a novel approach to dynamic estimation of rater reliability was proposed and evaluated, using both simulated and real-life experiments. This approach, Dynamic Estimation of Rater Reliability in Regression (DER^3) does not suffer from many of the limitations of state-of-the-art dynamic approaches. For instance, it is suited to a broad variety of tasks, can be used in the scenario where raters are not permanently engaged in the rating process, and does not require any prior knowledge about the task such as the statistical distribution of rater errors. The results of the experiments show that DER^3 can be used to decrease the error of predictions, keeping costs and time low.

The remainder of this chapter summarises how this thesis contributes toward crowdsourcing research and suggests areas that are worthy of future study.

9.1 Contributions

The two main contributions of the thesis are the following:

- **DER^3, a novel approach to dynamic estimation of rater reliability in the scenario of intermittent rater availability (Chapter 7):** many modern dynamic algorithms for the estimation of rater reliability have numerous problems when applied to real-life tasks. For instance, they are often suited only to a narrow category of rating tasks, e.g. binary classification. Often, these algorithms also make unreasonable assumptions, for instance, a large amount of prior knowledge about the task, such as knowing statistical distributions of rater errors. DER^3 is based on multi-armed bandits, and is specifically suited for practical applications, and is free of many such assumptions. The DER^3 (Dynamic Estimation of Rater Reliability for Regression) approach is designed for tasks where raters are not permanently engaged in the rating process. It was shown that this approach demonstrated significantly better results in simulated experiments, compared to the First-come-first-served approach that accepts ratings from all raters that are available without performing any estimation of their reliability.

- **Real-life evaluation of dynamic estimation of rater reliability based on multi-armed bandits (Chapter 8):** in order to perform additional validation of DER^3, it was implemented on the Amazon Mechanical Turk platform to gather rating for an in-house emotional speech corpus. It was validated that DER^3 can be used in practice and delivers results of a higher accuracy than the typical Amazon Mechanical Turk approach, where any rater is allowed to rate as many instances as desired and the instances are presented in random order.

Additionally, the following supporting contributions were made in the thesis:

- **An approach to dynamic estimation of rater reliability in the scenario of constant rater availability (Chapter 5):** in order to check the feasibility of using multi-armed bandits for dynamic estimation of rater reliability, an approach for the scenario of constant availability was developed. Two multi-armed bandit algorithms were used: ϵ-first and KL-UCB. Their performance, both in terms of cost and error of predictions, was compared to a few baselines. One of them was

asking random raters that corresponds to a scenario where no quality control is carried out at all. Another baseline was the Best Overall approach that asked the best raters which were known in advance. The Best Overall approach represented the best performance that can be achieved from a set of raters. The third baseline was IEThresh, a state-of-the-art algorithm for performing dynamic estimation in the conditions of constant rater availability. The comparisons resulted in the following conclusions:

  - Even a very simple ϵ-first technique can reach results of the same or higher quality as IEThresh baseline, using significantly fewer raters.
  - Both KL-UCB and ϵ-first are capable of choosing accurate raters well, and decreasing the total cost of the rating process.

- **An investigation into handling the bootstrap issue in the scenario of constant rater availability (Chapter 6):** one of the problems with using multi-armed bandits is that they conduct exploration in the beginning of the rating process. At this stage, rater reliabilities have not yet been precisely estimated, and as a result, noisy raters are often asked. This means that a certain portion of training instances usually receive a big number of noisy ratings and, therefore, predictions for them are likely to be noisy as well. The proposed algorithm consisted of trying to decrease the error of predictions by detecting the border between exploration and exploitation at the end of the rating process. Raters who proved to be reliable at the end were asked to submit ratings for the instances rated at the exploration phase. The results suggest that when ϵ-first is used to estimate rater reliability it is a trivial task to find the border between exploration and exploitation, so it is easy to decrease the error of prediction. Collecting additional ratings also proves worthwhile considering the additional costs associated with collection of additional ratings. However, in KL-UCB the detection of this border was much more difficult. This is not unexpected, since KL-UCB does not make a clear distinction between exploration and exploitation. Therefore, we were unable to devise an approach to solving the bootstrap issue for

KL-UCB.

- **Contextualisation of dynamic approaches (Chapter 3)** a variety of static and dynamic techniques exist; however, researchers rarely remark that a static or dynamic technique may be preferable, depending on the task at hand. In order to better contextualise dynamic approaches with respect to estimation of rater reliability, an analysis of the characteristics of rating tasks was carried out. Taxonomies of currently available crowdsourcing tasks were reviewed, and some of the applicable characteristics/dimensions were re-used. However, these dimensions were not sufficient to make a proper distinction between rating tasks. Several more characteristics of rating tasks were identified. The conclusion of this review is that dynamic approaches are especially useful in the tasks where the cost of a single rating is high, as well as for tasks where providing a single rating takes a long time.

- **Benchmark of supervised classification techniques on emotional speech data (Appendix A):** Since there is no consensus on which classification techniques are best in the domain of emotion recognition from speech, multiple algorithms were tested on four natural emotional speech datasets. The following techniques were compared:

  1. Support Vector Machine with linear kernel
  2. Support Vector Machine with radial basis function kernel
  3. C4.5 decision tree
  4. k nearest neighbour algorithm
  5. Radial basis function neural network
  6. Naive Bayes classificator
  7. Multi-layered perceptron neural network

  Application of Friedman and Holm tests concluded that Support Vector Machines showed significantly better performance than any other technique. These results were used to select a predictor to be used in active learning, a technique applied

to choosing the sequence of instances to be presented to raters, in experiments conducted in this thesis.

9.2 Future work

The contributions listed in the previous section are centred around the problem of dynamic estimation of rater reliability. However, the exploration of the problem unearthed more questions that can be addressed in one thesis. Some potentially interesting areas for future research include the following questions:

1. **How well the performance of DER^3 scales when the number of raters and/or training instances is of the order of hundreds or thousands?** Experiments in this thesis usually operated with crowdsourcing datasets where the number of raters never exceeded twenty-seven and the number of instances was of the order of hundreds. It might be worthwhile to check whether DER^3 still performs well for tasks with much bigger numbers of raters and training instances and, potentially, suggest some changes to the algorithm that improve its performance under such conditions.

2. **How does the performance of DER^3 change if the importance of time, cost and error are not equally weighted?** In this thesis we assumed that cost, error and time are equally important, when choosing the best approach to dynamic estimation of rater reliability. However, there can be some practical tasks where this might not be true, for instance, a supervised machine learning task where it is known that a certain classifier can cope well with noise in training data. In this case the importance of error is going to be lower than that of cost and time. Also, in many cases it might be difficult to specify the weights exactly. For example, paying less may be more important than getting ratings in a short time, but it may be difficult to express it in precise numeric terms. It seems to be worthwhile to explore such cases of non-trivial weighting schemes following the work by Soliman et al. [167] (and references within).

3. **Is it possible to collect a different number of ratings per instance in the scenario of dynamic estimation of rater reliability?** In all our experiments we aimed to collect N ratings per training instance. However, some algorithms were able to achieve low error collecting less than N ratings. We believe that it is worthwhile to consider the dynamic selection of N as an additional cost-saving measure, and to suggest and evaluate approaches to do it in the DER^3 approach.

4. **How to determine the order of instance presentation when no features are available?** In this thesis, when no features were available in a dataset, the sequence of presentation was chosen randomly. However, more sophisticated approaches (e.g. using information about ratings collected to date) can be used. Instance presentation based on the degree of consensus seems promising, i.e. we first seek ratings for those instances where raters tend to disagree with each other.

5. **Is it possible to encourage raters to come to rate more often in DER^3/ϵ-first* and DER^3/KL-UCB*?** These two approaches achieved very low error, because they applied quite severe filtering of raters. The drawback was that the rating process became long, as only a small number of raters was allowed to rate. Potentially, raters can be motivated to come back more often if they, for instance, are offered a higher price per rating. At the same time, such additional costs might not be justified by the decreased prediction error. It would be interesting to conduct a simulated experiment to see if such motivation schemas are actually worthwhile in the context of DER^3.

6. **What are the limits of the DER^3 approach, when it is applied to real-life tasks?** The real-life evaluation of DER^3 strongly suggests that this approach can be very beneficial on a real-life crowdsourcing platform. However, it would be interesting to see how this approach behaves on tasks from other areas with different characteristics. For instance, binary or multi-class classification problems, problems with a high degree of subjectivity, problems where a very high number of ratings is required for each training instance, etc. It might also be worth investigating whether

DER^3 can be used in crowdsourcing tasks other than rating corpora for supervised machine learning.

7. **Is it possible to improve the performance of the DER^3 approach by modelling variation in the performance of a rater due to fatigue or learning?** The performance of a rater might not be the same at all times when ratings are being submitted. For instance, a rater might become more experienced, and as a result, the quality of his work might improve. Conversely, a rater may become tired or inattentive, and start making more mistakes. Although incorporation of such effects can be approached in different ways, we would suggest investigating the use of Markovian multi-armed bandits (Section 3.5), where different internal states correspond to different degrees of rater efficiency, for instance, "Tired", "Normal", "Inexperienced", "Very experienced" and so on.

Appendix A

Comparison of classification techniques for emotion recognition from speech

In the field of emotion recognition from speech there is no consensus about which classifier/predictor is the most accurate as the detailed comparison of techniques across variety of datasets is usually not the focus of the state-of-the-art research in this area. In order to choose the technique for active learning, we conducted our own experiment, comparing the accuracy of several machine learning algorithms on four natural emotional speech corpora. We used the following machine learning algorithms as the most widely used in this domain:

1. Support Vector Machine with linear kernel (SVM-Linear).
2. Support Vector Machine with radial basis function kernel (SVM-RBF).
3. C4.5 decision tree (C4.5).
4. k nearest neighbour algorithm (k-NN).
5. Radial basis function neural network (RBF).
6. Naive Bayes classificator (NB).
7. Multi-layered perceptron neural network (MLP).

The following natural emotional speech corpora were used:

1. **FAU Aibo Emotion Corpus**—this corpus [18, 170] contains recordings of children speaking to the AIBO robot controlled by a human invisible to them. All 18,216 instances in the corpus were used in the Interspeech 2009 Challenge in a five-class classification problem (angry, emphatic, neutral, positive, rest) and the datasets extracted from this corpus used this labelling scheme. Instances with a high confidence level[1] were selected where possible (above 0.5 on a 0 to 1 scale).

   Four datasets were extracted from this corpus—300 confident neutral instances and 300 angry instances were randomly selected and formed the first dataset labelled as AIBO-NA. The second and third datasets, labelled AIBO-NE and AIBO-NP, were generated in the same way but contained 300 emphatic instances and 300 positive instances respectively, in place of the angry instances. The fourth dataset, AIBO-AENPR involved selecting 200 random instances from each of the five categories. In all cases, except Rest where there were few instances with high confidence, these were confident instances.

   An additional dataset was derived from the 4-class corpus proposed by the HUMAINE Network of Excellence CEICES initiative that contained 4,513 instances from the AIBO corpus. The same procedure for selecting instances was applied, creating a dataset containing 200 confident instances from each class.

2. **BabyEars**—this corpus contains recordings of parents speaking to their children [162]. All instances belong to one of three classes: approval (when the action of a child is approved), attention (when the parent attracts the attention of the child) and prohibition (when parents prohibit some actions). This corpus consists of 509 instances.

3. **Vera am Mittag German Audio-Visual Emotional Speech Database** is a natural corpus of German talk-show recordings [74] which contains instances rated on three dimensional scales on a scale of –1 to +1. The dimensions were activa-

[1]The AIBO corpus was labelled at the word level, and these labels were used to get a label for the whole phrase. The confidence level for the utterance denotes the proportion of words from that utterance that have the same label as the utterance itself.

tion, valence (a synonym for evaluation) and dominance (a synonym for power). A single dataset of three classes was generated from the ratings across each of these dimensions, VAM-ACT, VAM-EVAL and VAM-POW. Each dataset consists of 947 instances.

4. **Utsunomiya University Spoken Dialogue Database For Paralinguistic Information Studies**[2]—a Japanese elicited corpus that contains 4,840 instances labelled across six dimensions (pleasantness, arousal, dominance, credibility, interest and positivity) on a scale of 1 to 7. Each instance in this corpus has rating values supplied by three experts and the mean of these values is used as the target rating for each instance. For the purposes of dataset generation for this study a measure of confidence was assigned to each instance where confidence was measured as the difference between the minimal and maximal rating given to each instance.

   Only the ratings from the dimensions of arousal (activation), pleasantness (evaluation) and dominance (power) were used. For each dimension the instances were discretized into three classes in the manner described below. Where classes contained more than 300 instances having the same value of the measure of confidence, a random selection was chosen to generate the datasets labelled UUDB-ACT, UUDB-EVAL and UUDB-POW reflecting the arousal, pleasantness and dominance dimensions.

Table A.1: Discretisation of the dimensional datasets

Dataset	Lower class		Middle class		Higher class	
	Instances	Range	Instances	Range	Instances	Range
UUDB-ACT	1496	[1; 3.6667]	1619	[4; 4.6667]	1725	[5; 7]
UUDB-DOM	1484	[1; 3]	1564	[3.3333; 4.6667]	1792	[5; 7]
UUDB-VAL	1847	[1; 3.6667]	1753	[4; 4.3333]	1240	[4.6667; 7]
VAM-ACT	317	[-1; -0.1625]	315	[-0.1586; 0.1169]	315	[0.1172; 1]
VAM-DOM	317	[-1; -0.0562]	315	[-0.0546; 0.1852]	315	[0.1853; 1]
VAM-VAL	317	[-1; -0.293]	315	[-0.2892; -0.163]	315	[-0.1623; 1]

The VAM and UUDB databases contain ratings on continuous scales, however, the task at hand is classification. In order to use classification algorithms on VAM and UUDB and

[2]http://uudb.speech-lab.org/

to compare the accuracies of machine learning techniques across all datasets, VAM and UUDB ratings were converted to ordinal scales. Instances were sorted within a particular dimension and then separated into three classes in such a way that each discrete class contains approximately the same number of instances. Table A.1 provides the number of instances in each class created in this way and the range of values assigned to each class. The details of all datasets are given in Table A.2.

Table A.2: Datasets used

Name	Language	Description of classes	Number of instances	Number of classes	Data distribution
AIBO-AENPR	German	Anger, empathy, neutral, positive, rest	1000	5	20/20/20/20/20
AIBO-AMEN	German	Anger, motherese, empathy, neutral	800	4	25/25/25/25
AIBO-NA	German	Neutral, anger	600	2	50/50
AIBO-NE	German	Neutral, empathic	600	2	50/50
AIBO-NP	German	Neutral, positive	600	2	50/50
BabyEars	English	Attention, approval, prohibition	509	3	42/29/29
UUDB-ACT	Japanese	Three levels of activation	900	3	33/33/33
UUDB-EVAL	Japanese	Three levels of valence	900	3	33/33/33
UUDB-POW	Japanese	Three levels of dominance	900	3	33/33/33
VAM-ACT	German	Three levels of activation	947	3	33/33/33
VAM-EVAL	German	Three levels of valence	947	3	33/33/33
VAM-POW	German	Three levels of dominance	947	3	33/33/33

For the experiments a feature set of 384 acoustical and spectral features was extracted using openEAR software[3]. This feature set was used in the Interspeech 2009 Challenge and includes features based on pitch, energy, zero-crossing rate, harmonics to noise ratio as well as 12 mel-frequency cepstral coefficients. All features were normalised to the interval [0; 1] with the only exception in the case of MLP and RBF where the interval [-1; 1] was used as it is best-practice for the back-propagation algorithm allowing it to converge faster.

Each classifier was evaluated on each dataset using 5-fold cross validation. At each iteration of the cross-validation, the parameters of each classifier (except NB which does not have any parameters) were tuned on the training set using an additional 5-fold cross validation. The ranges used to tune the parameters are detailed in Table A.3. The performance measure used was average class accuracy as a number of the datasets were imbalanced. This overall process was repeated three times and averaged classification

[3]http://openart.sourceforge.net/

accuracies for each classifier on each dataset are calculated.

Table A.3: The ranges of values used in parameter tuning.

Classifier	**Parameter**	**Ranges Tested**
C4.5	Confidence threshold for pruning	$[0.05, 0.1, ..., 0.5]$
	Minimum number of instances per leaf	$[1, 2, 3, 4]$
k-NN	Number of nearest neighbours	$[1, 3, ..., 41]$
MLP	Number of hidden neurons	$[100, 200, 300]$
RBF	Number of hidden neurons	$[2, 3, ..., 10]$
SVM-LIN	Cost parameter, C	$[2^{-5}, 2^{-4}, ..., 2^{15}]$
SVM-RBF	Cost parameter, C	$[2^{-5}, 2^{-4}, ..., 2^{15}]$
	Kernel parameter, γ	$[2^{-15}, 2^{-14}, ..., 2^{3}]$

Table A.4 presents the average classification accuracies for each classifier on each dataset and includes the average ranks of classifiers. To see if there is a statistically signifiant difference between their performance, we used a two-stage procedure suggested by Demsar [49]: first, we used Friedman test to see if there is any difference between any approaches, second, we conducted a post-hoc Holm test to check which algorithms are not significantly different compared to the best technique. Friedman test showed that there is a statistically significant difference between at least some techniques (p-value=2×10^{-10}). According to Holm test, the performance of the best technique, SVM-RBF, is not significantly different from that of SVM-Linear and MLP.

Table A.4: Results of the comparison of classifiers: average classification accuracies (A) and ranks (R).

Dataset	SVM-Linear		SVM-RBF		C4.5		k-NN		RBF		NB		MLP	
	A	**R**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**
AIBO-AENPR	49.00	2	50.20	1	34.40	7	45.50	4	41.10	6	42.30	5	45.89	3
AIBO-AMEN	63.98	2	66.46	1	46.67	7	61.00	3	54.67	5	54.00	6	60.50	4
AIBO-NA	81.00	1	80.50	2	71.44	6	73.83	4	72.61	5	60.67	7	78.60	3
AIBO-NE	79.33	3	80.11	1	73.17	5	77.17	4	71.28	6	64.50	7	79.51	2
AIBO-NP	76.53	2	77.54	1	68.46	7	74.61	3	72.93	6	73.38	5	74.36	4
BabyEars	70.55	2	77.56	1	55.56	7	64.27	4	56.84	5	56.80	6	68.91	3
UUDB-ACT	75.00	2	75.37	1	69.59	5	69.67	4	68.48	6	66.89	7	73.36	3
UUDB-EVAL	60.74	1	59.37	2	51.15	6	53.30	4	52.44	5	47.89	7	55.87	3
UUDB-POW	75.11	1	74.19	2	66.04	7	71.18	3	69.63	6	71.00	4	70.56	5
VAM-ACT	62.80	1	61.96	2	52.62	7	55.77	5	53.02	6	56.69	4	57.78	3
VAM-EVAL	45.64	4	49.40	1	40.98	7	46.59	2	43.14	6	46.13	3	45.07	5
VAM-POW	56.81	4	70.08	1	54.10	5	65.21	3	50.57	6	46.92	7	65.33	2
Averaged rank	2.08		1.33		6.33		3.58		5.67		5.67		3.33	

Appendix B

Development and testing of the rating interface

To rate instances on Amazon Mechanical Turk, a dedicated rating interface was developed. In order to evaluate this interface, we recruited 7 volunteers, none of which was an expert in emotional speech recognition. The rating interface was implemented as a separate rating tool to aid the validation process. We were interested in two aspects of the rating tool:

1. Do raters understand the terms of activation and evaluation?

2. Is the interface convenient enough?

Both aspects are critical for the successful rating process. If the tool is user friendly, but instruction do not make sense, raters would not be able to rate activation and evaluation as they might have difficulties with these concepts. But even if instructions are crystal clear, the interface should be user-friendly to make the process easy and enjoyable. We checked both these aspects separately: first, each volunteer had to read the instructions and then answer the following multiple-choice questions about the definitions of Activation and Evaluation (correct answers are given in bold):

1. Which of the following is best described by Evaluation (pick one answer):

   (a) A speech segment relating to examination.

   (b) A speech segment that sounds like a whisper.

(c) **A speech segment where the speakers voice conveys the benefit of (or problem with) something.**

(d) A speech segment that indicates the age of the person.

2. Which of the following is best described by activation (pick one answer):

(a) A speech segment relating to work levels.

(b) A speech segment relating to politics.

(c) **A speech segment that contains physical arousal in the voice due to emotion.**

(d) A speech segment where the speaker begins an action.

Second, the volunteers rated a few instances and performed a self-evaluation of workload using an adapted NASA Task Load Index (TLX) questionnaire[1]. The original NASA TLX questionnaire was designed to assess workload associated with the working with a particular human-machine system. In its original form it asks to perform a self-assessment on five scales:

1. Mental demand: how mentally demanding was the task?
2. Physical demand: how physically demanding was the task?
3. Temporal demand: how hurried or rushed was the pace of the task?
4. Performance: how successful were you in accomplishing what you were asked to do?
5. Effort: how hard did you have to work to accomplish your level of performance?
6. Frustration: how insecure, discouraged, irritated, stressed and annoyed were you?

The rating task did not involve any physical work, so the question #2 was not asked. We also excluded question #4 as there was no immediately visible and measurable result of rating accuracy. As the task at hand was quick, easy and risk-free, we decided not to ask question #6 either. Another change that we made was alteration of self-assessment

[1] http://humansystems.arc.nasa.gov/groups/tlx/

Table B.1: Results of volunteers' self-assessment after working with the rating interface.

	Very Low	Low	Normal	High	Very High
Mental demand	1	1	3	2	0
Temporal demand	0	3	4	0	0
Effort	3	1	2	1	0

scales. In order to make the self-assessment easier, we reduced the original NASA TLX 21-point scales to six point-scales, from *Very Low* to *Very High*.

Six people out of seven were able to successfully answer the questions about activation and evaluation which strongly suggests that they understood the instructions well. The self-assessment results are presented in Table B.1. As can be seen, none of the volunteers considered that the task required very high demand on any scale. There were a few people for whom the task seemed to be quite demanding as they rated it as "High" on some scales, but it happened quite rarely. Most participants thought that the effort required was normal or even low, which lets us to conclude that the rating interface is suitable for rating emotional speech.

Appendix C

Implementation of DER^3 on Amazon Mechanical Turk

In order to additionally validate DER^3, it was implemented on Amazon Mechanical Turk platform in the following way. There is a special type of AMT tasks called *ExternalHIT* (external human intelligence task). When an ExternalHIT is used, the contents of an external web page are displayed instead of the default AMT rating interface. A typical use case for such HITs is a task where the rating interface is highly specific and is not available among the AMT templates. However, an ExternalHIT can also be used to facilitate advanced processes such as the DER^3 approach.

The implementation of an ExternalHIT in our experiments is summarised in Figure C.1. When a worker becomes available to provide a rating and accepts a task, his unique ID is transferred to the `klucb.php` script as a `POST` parameter, as happens with any ExternalHIT. The page to which the ID is sent is typically responsible for displaying the rating interface, collecting a rating and then notifying AMT that a rater completed a task so he can get paid. In our implementation these responsibilities are shared between the `klucb.php` and `update.php` scripts. The `klucb.php` script is connected to a database that stores all ratings gathered to date. This is used to check if there are any instances that the rater can rate. There are two possibilities: (i) there are no suitable instances, in which case a message asking the rater to come back later is described. And (ii) there are instances to be rated, in which case a single instance is selected and presented to the

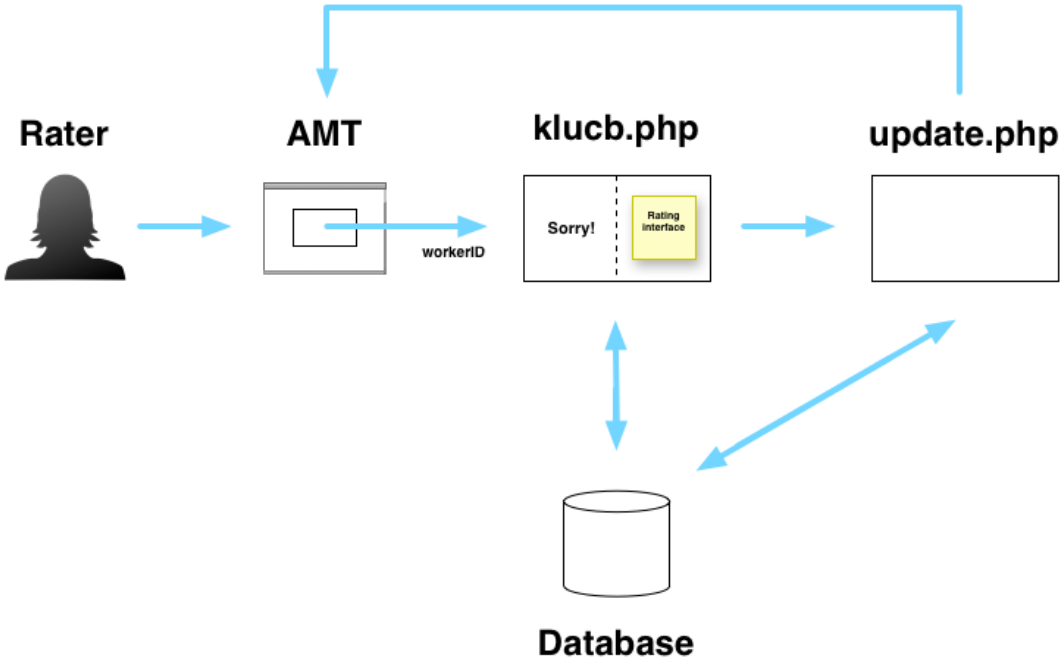


Figure C.1: Overview of the AMT implementation of DER[3].

rater. When the rater submits a rating, it is forwarded to the `update.php` script. The `update.php` script updates the database with a new rating, as well as informs AMT that the rater successfully completed a task. Each HIT consisted of rating a recording only on one emotional dimension.

Appendix D

Statistical Tests and Procedures used in the Thesis

D.1 Friedman test

Friedman test [49] checks whether there is a statistically significant difference between N approaches each of which was used on k datasets. Each approach is characterised by its average rank $R_j, j = (1, 2, ..., N)$. The null hypothesis is that there is no significant difference between any approaches, i.e. that all of them are equivalent. The Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \tag{D.1}$$

is distributed according to χ_F^2 distribution with $k-1$ degrees of freedom.

D.2 Holm procedure

Holm procedure [49] works with a list of N approaches ranked by their ranks $R_j, j = (1, 2, ..., N)$. It checks whether there is a statistically significant difference between the rank of the best approach and every other approach, assuming that all approaches were tested on k datasets. In order to do it, a z-statistic is calculated for each $i = 2, 3, ..., N$ approach in the following way:

$$z = \frac{R_1 - R_i}{\sqrt{\frac{k(k+1)}{6N}}}. \tag{D.2}$$

This statistic is distributed as a standard normal distribution. For each approach a corresponding p-value p_i is calculated. Approach with a number i is not significantly different from the approach that ranked first if $p_i < \alpha/(k-i)$, where α is significance level.

D.3 Bergmann-Hommel procedure

Bergmann-Hommel procedure [64] splits N algorithms into groups in such a way that performance of the algorithms from the same group is not significantly different from each other. The z-scores are calculated using formula D.2 (as well as p-values), but for all possible pairs of N algorithms. Then a number of hypotheses is obtained. For instance, if $N = 3$, hypotheses are the following:

- H_1: Algorithm #1 has the same performance as Algorithm #2.
- H_2: Algorithm #1 has the same performance as Algorithm #3.
- H_3: Algorithm #2 has the same performance as Algorithm #3.

Every hypothesis can be true or false. Overall, there are 2^N sets of all possible combinations of true and false hypotheses. For $N = 3$ there are eight such sets:

- S_1: H_1, H_2 and H_3 are true.
- S_2: H_1 and H_2 are true, H_3 is false.
- S_3: H_1 and H_3 are true, H_2 is false.
- S_4: H_2 and H_3 are true, H_1 is false.
- S_5: H_1 is true, H_2 and H_3 are false.
- S_5: H_1 is true, H_2 and H_3 are false.
- S_6: H_2 is true, H_1 and H_3 are false.
- S_7: H_3 is true, H_1 and H_2 are false.
- S_8: H_1, H_2 and H_3 are false.

Some of these sets are not possible, basing on the laws of logics (for instance, S_2). All sets except such "impossible" sets are called *exhaustive* sets.

Bergmann-Hommel procedure rejects all H_j with $j \notin A$ where

$$A = \cup \{I : I \text{ is an exhaustive set}, \min \{p_i : i \in I\} > \alpha/|I|\}. \tag{D.3}$$

Typically, the set of retained hypotheses allows to divide all algorithms into groups by their performance. For more details about this procedure refer to the work by Garcia and Herrera [64].

D.4 Mann-Kendall test

The purpose of Mann-Kendall test [117] is to check whether the sequence of numbers $x_1, x_2, ..., x_M$ has a negative or positive trend. The null hypothesis is that all observations are independent and identically distributed, i.e. there is no upward or downward trend in the data. Test statistic

$$S = \sum_{i<k} sign(x_k - x_i), \tag{D.4}$$

where

$$sign(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases} \tag{D.5}$$

is normally distributed with parameters $\mu = 0$ and $\sigma = \frac{M(M-1)(2M+5)}{18}$.

Bibliography

[1] Ahn, L. and Dabbish, L.: 2004, Labeling Images with a Computer Game, *Proceedings of CHI*, pp. 319–326.

[2] Ai, H., Litman, D., Forbes-Riley, K., Rotaru, M., Tetreault, J. and Purandare, A.: 2006, Using System and User Performance Features to Improve Emotion Detection in Spoken Tutoring Dialogs, *Proceedings of the Ninth International Conference on Spoken Language Processing*.

[3] Alpaydin, E.: 2010, *Introduction to Machine Learning*, 2nd edn, MIT Press.

[4] Altun, H. and Polat, G.: 2007, New Frameworks to Boost Feature Selection Algorithms in Emotion Detection for Improved Human-Computer Interaction, *LNCS, BVAI 2007* **4729**, 533–541.

[5] Alvarez, A., Cearetta, I., Lopez, J., Arruti, A., Lazkano, E., Sierra, B. and Garay, N.: 2007, A Comparison Using Different Speech Parameters in the Automatic Emotion Recognition Using Feature Subset Selection Based on Evolutionary Algorithms, *LNAI, TSD 2007* **4629**, 423–430.

[6] Alvarez, A., Cearreta, I., Lopez, J., Arruti, A., Lazkano, E., Sierra, B. and Garay, N.: 2006, Feature Subset Selection Based on Evolutionary Algorithms for Automatic Emotion Recognition in Spoken Spanish and Standard Basque Language, *LNAI, TSD 2006* **4188**, 565–572.

[7] Alvarez, A., Cearreta, I., Lopez, J., Arruti, A., Lazkano, E., Sierra, B. and Garay,

N.: 2007, Application of Feature Subset Selection Based on Evolutionary Algorithms for Automatic Emotion Recognition in Speech, *LNAI, NOLISP 2007* **4885**, 273–281.

[8] Ambati, V., Vogel, S. and Carbonell, J.: 2010, Active Learning and Crowd-sourcing for Machine Translation, *Proceedings of LREC*.

[9] Artstein, R. and Poesio, M.: 2005, Kappa 3 = Alpha (or Beta), *Technical Report CSM-437*, Natural Language Engineering and Web Applications Group, Department of Computer Science, University of Essex.

[10] Athanaselis, T., Bakamidis, S., Dologlou, I., Cowie, R., Douglas-Cowie, E. and Cox, C.: 2005, ASR for Emotional Speech: Clarifying the Issues and Enhancing Performance, *Neural Networks* **18**, 437–444.

[11] Audhkhasi, K. and Narayanan, S.: 2013, A Globally-Variant Locally-Constant Model for Fusion of Labels from Multiple Diverse Experts without Using Reference Labels, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35**(4), 769–783.

[12] Audibert, J. and Bubeck, S.: 2010, Regret Bounds and Minimax Policies Under Partial Monitoring, *The Journal of Machine Learning Research* **11**, 2785–2836.

[13] Audibert, J., Munos, R. and Szepesvari, C.: 2009, Exploration-exploitation Tradeoff using Variance Estimates in Multi-armed Bandits, *Theoretical Computer Science* **410**(19), 1876–1902.

[14] Auer, P., Cesa-Bianchi, N., Freund, Y. and Schapire, R. E.: 2003, The nonstochastic multiarmed bandit problem, *SIAM Journal on Computing* **32**(1), 48–77.

[15] Bachrach, Y., Minka, T., Guiver, J. and Graepel, T.: 2012, How To Grade a Test Without Knowing the Answers — A Bayesian Graphical Model for Adaptive Crowdsourcing and Aptitude Testing, *Proceedings of ICML*.

[16] Batliner, A., Fischer, K., Huber, R., Spilker, J. and Nöth, E.: 2003, How to Find Trouble in Communication, *Speech Communication* **40**(1-2), 117–143.

[17] Batliner, A., Hacker, C., Steidl, S., Nöth, E., D'Arcy, S., Russell, M. and Wong, M.: 2004, 'You stupid tin box'—Children Interacting with the AIBO Robot: A Cross-linguistic Emotional Speech Corpus, *Proceedings of LREC*, pp. 171–174.

[18] Batliner, A., Steidl, S., Hacker, C. and Nöth, E.: 2008, Private Emotions versus Social Interaction: A Data-driven Approach towards Analysing Emotion in Speech, *User Modeling and User-Adapted Interaction* **18**, 175–206.

[19] Batliner, A., Steidl, S., Hacker, C., Nöth, E. and Niemann, H.: 2005, Tales of Tuning—Prototyping for Automatic Classification of Emotional User States, *Proceedings of INTERSPEECH*.

[20] Batliner, A., Steidl, S., Schuller, B., Seppi, D., Laskowski, K., Vogt, T., Devillers, L., Vidrascu, L., Kessous, L. and Aharonson, V.: 2006, Combining Efforts for Improving Automatic Classification of Emotional User States, *Proceedings of the Fifth Slovenian and First International Language Technologies Conference* pp. 240–245.

[21] Batliner, A., Steidl, S., Schuller, B., Seppi, D., Vogt, T., Wagner, J., Devillers, L., Vidrascu, L., Aharonson, V., Kessous, L. and Amir, N.: 2010, Whodunnit — Searching for the Most Important Feature Types Signalling Emotion-related User States in Speech, *Computer Speech and Language* **25**(1), 4–28.

[22] Brew, A., Greene, D. and Cunningham, P.: 2010a, Is it Over Yet? Learning to Recognize Good News in Financial Media, *Technical Report UCD-CSI-2010-1*, University College Dublin.

[23] Brew, A., Greene, D. and Cunningham, P.: 2010b, Using Crowdsourcing and Active Learning to Track Sentiment in Online Media, *Proceedings of ECAI*.

[24] Bubeck, S. and Cesa-Bianchi, N.: 2012, Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems, *Foundations and Trends in Machine Learning* **5**(1), 1–122.

[25] Burbidge, R., Rowland, J. and King, R.: 2007, Active Learning for Regression Based on Query by Committee, *LNCS, IDEAL 2007* **4881**, 209–218.

[26] Burkhardt, F., Ajmera, J., Englert, R., Stegmann, J. and Burleson, W.: 2006, Detecting Anger in Automated Voice Portal Dialogs, *Proceedings of INTERSPEECH*.

[27] Burkhardt, F., Paeschke, A., Rolfes, M., Sendlmeier, W. and Weiss, B.: 2005, A Database of German Emotional Speech, *Proceedings of INTERSPEECH*.

[28] Busso, C., Lee, S. and Narayanan, S.: 2007, Using Neutral Speech Models for Emotional Speech Analysis, *Proceedings of INTERSPEECH*, pp. 2225–2228.

[29] Caelen, O. and Bontempi, G.: 2010, A Dynamic Programming Strategy to Balance Exploration and Exploitation in the Bandit Problem, *Annals of Mathematics and Artificial Intelligence* **60**(1), 3–24.

[30] Callejas, Z. and Lopez-Cozar, R.: 2008, Influence of Contextual Information in Emotion Annotation for Spoken Dialogue Systems, *Speech Communication* **50**, 416–433.

[31] Callison-Burch, C.: 2009, Fast, Cheap, and Creative: Evaluating Translation Quality using Amazon's Mechanical Turk, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[32] Carletta, J.: 1996, Assessing Agreement on Classification Tasks: the Kappa Statistic, *Computational Linguistics* **22**(2).

[33] Chen, X., Lin, Q. and Zhou, D.: 2013, Optimistic Knowledge Gradient Policy for Optimal Budget Allocation in Crowdsourcing, *Proceedings of ICML*.

[34] Chien-Ju Ho, J. W. V.: 2012, Online Task Assignment in Crowdsourcing Markets, *Proceedings of AAAI*.

[35] Cho, J., Kato, S. and Itoh, H.: 2008, A Biphase-Bayesian-Based Method of Emotion Detection from Talking Voice, *LNAI, KES 2008, Part III* **5179**, 50–57.

[36] Cholleti, S., Goldman, S., Blum, A., Politte, D. and Don, S.: 2008, Veritas: Combining Expert Opinions without Labeled Data, *Technical report*.

[37] Cohen, J.: 1960, A Coefficient of Agreement for Nominal Scales, *Educational and Psychological Measurement* **20**, 37–46.

[38] Corney, J., Torres-Sanchez, C., Jagadeesan, A. and Regli, W.: 2009, Outsorcing Labour to the Cloud, *International Journal of Innovation and Sustainable Development* **4**(4), 294–313.

[39] Cowie, R. and Cornelius, R.: 2003, Describing the Emotional States that are Expressed in Speech, *Speech Communication* **40**, 5–32.

[40] Cowie, R., Douglas-Cowie, E. and Cox, C.: 2005, Beyond Emotion Archetypes: Databases for Emotion Modelling using Neural Networks, *Neural Networks* **18**, 371–388.

[41] Craggs, R.: 2004, Annotating Emotion in Dialogue: Issues and Approaches, *Proceedings of CLUK Research Colloquium*.

[42] Dai, P., Lin, C. H., Mausam and Weld, D. S.: 2013, POMDP-based Control of Workflows for Crowdsourcing, *Artificial Intelligence* **202**(C), 52–85.

[43] Dalvi, N., Dasgupta, A., Kumar, R. and Rastogi, V.: 2013, Aggregating Crowdsourced Binary Ratings, *Proceedings of WWW*.

[44] Danisman, T. and Alpkocak, A.: 2008, Emotion Classification of Audio Signals Using Ensemble of Support Vector Machines, *LNAI, PIT 2008* **5078**, 205–216.

[45] Darwin, C.: 1872, *The Expression of Emotions in Man and Animals*, Albermarle.

[46] Dawid, A. and Skene, A.: 1979, Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm, *Journal of the Royal Statistical Society. Series C (Applied Statistics)* pp. 20–28.

[47] Dekel, O., Gentile, C. and Sridharan, K.: 2012, Selective Sampling and Active Learning from Single and Multiple Teachers, *The Journal of Machine Learning Research* **13**(1).

[48] Dekel, O. and Shamir, O.: 2009, Good Learners for Evil Teachers, *Proceedings of ICML*.

[49] Demsar, J.: 2006, Statistical Comparisons of Classifiers over Multiple Data Sets, *Journal of Machine Learning Research* **7**.

[50] Devillers, L., Cowie, R., Martin, J., Douglas-Cowie, E., Abrilian, S. and McRorie, M.: 2006, Real Life Emotions in French and English TV Video Clips: an Integrated Annotation Protocol Combining Continuous and Discrete Approaches, *Proceedings of LREC*, pp. 1105–1110.

[51] Devillers, L. and Vidrascu, L.: 2007, Real-Life Emotion Recognition in Speech, *LNAI, Speaker Classification II* **4441**, 34–42.

[52] Devillers, L., Vidrascu, L. and Lamel, L.: 2005, Challenges in Real-life Emotion Annotation and Machine Learning based Detection, *Neural Networks* **18**(4), 407–422.

[53] D'Mello, S., Craig, S., Witherspoon, A., McDaniel, B. and Graesser, A.: 2008, Automatic Detection of Learner's Affect from Conversational Cues, *User Modeling and User-Adapted Interaction* **18**(1-2), 45–80.

[54] Doan, A., Ramakrishnan, R. and Halevy, A.: 2011, Crowdsourcing Systems on the World-Wide Web, *Communications of the ACM* **54**(4), 86–96.

[55] Donmez, P., Carbonell, J. and Schneider, J.: 2009, Efficiently Learning the Accuracy of Labeling Sources for Selective Sampling, *Proceedings of KDD*.

[56] Ekman, P., Friesen, W., O'Sullivan, M., Chan, A., Diacoyanni-Tarlatzis, I., Heider, K., Krause, R., LeCompte, W., Pitcairn, T. and Ricci-Bitti, P.: 1987, Universals and Cultural Differences in the Judgments of Facial Expressions of Emotion, *Journal of Personality and Social Psychology* **53**(4), 712–717.

[57] Erickson, T.: 2011, Some Thoughts on a Framework for Crowdsourcing, *Proceedings of Workshop on Crowdsourcing and Human Computation, in conjunction with CHI*.

[58] Ertekin, S., Hirsh, H. and Rudin, C.: 2011, Approximating the Wisdom of the Crowd, *Workshop on Computational Social Science and the Wisdom of Crowds, in conjunction with NIPS*.

[59] Estellés-Arolas, E. and Gonzalez-Ladron-de Guevara, F.: 2012, Towards an Integrated Crowdsourcing Definition, *Journal of Information Science* **38**(2), 189–200.

[60] Fang, M., Zhu, X., Li, B., Ding, W. and Wu, X.: 2012, Self-Taught Active Learning from Crowds, *Proceedings of ICDM*, pp. 858–863.

[61] Feng, D., Besana, S. and Zajac, R.: 2009, Acquiring High Quality Non-expert Knowledge from On-demand Workforce, *Proceedings of the Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*.

[62] Fersini, E., Messina, E., Arosio, G. and Archetti, F.: 2009, Audio-Based Emotion Recognition in Judicial Domain: A Multilayer Support Vector Machines Approach, *LNAI, MLDM 2009* **5632**, 594–602.

[63] Fischler, M. A. and Bolles, R. C.: 1981, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Commun. ACM* **24**(6), 381–395.

[64] Garcia, S. and Herrera, F.: 2008, An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons, *Journal of Machine Learning Research* **9**, 2677–2694.

[65] Garivier, E. and Cappe, O.: 2011, The KL-UCB Algorithm for Bounded Stochastic Bandits and Beyond, *Proceedings of COLT*.

[66] Geiger, D., Seedorf, S., Schulze, T., Nickerson, R. and Schader, M.: 2011, Managing the Crowd: Towards a Taxonomy of Crowdsourcing Processes, *Proceedings of the 17th Americas Conference on Information Systems*.

[67] Gittins, J.: 1979, Bandit Processes and Dynamic Allocation Indices, *Journal of the Royal Statistical Society. Series B (Methodological)* **41**(2), 148–177.

[68] Goldberg, K., Roeder, T., Gupta, D. and Perkins, C.: 2001, Eigentaste: A Constant Time Collaborative Filtering Algorithm, *Information Retrieval* **4**, 133–151.

[69] Grimm, M. and Kroschel, K.: 2005a, Evaluation of Natural Emotions using Self Assessment Manikins, *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 381–385.

[70] Grimm, M. and Kroschel, K.: 2005b, Rule-based Emotion Classification by Acoustic Features, *Proceedings of the Third International Conference on Telemedicine and Multimedia Communication*.

[71] Grimm, M., Kroschel, K., Harris, H., Nass, C., Schuller, B., Rigoll, G. and Moosmayr, T.: 2007, On the Necessity and Feasibility of Detecting a Driver's Emotional State While Driving, *LNCS, ACII 2007* **4738**, 126–138.

[72] Grimm, M., Kroschel, K., Mower, E. and Narayanan, S.: 2007, Primitives-based Evaluation and Estimation of Emotions in Speech, *Speech Communication* **49**, 787–800.

[73] Grimm, M., Kroschel, K. and Narayanan, S.: 2007, Support Vector Regression for Automatic Recognition of Spontaneous Emotions in Speech, *Proceedings of ICASSP*.

[74] Grimm, M., Kroschel, K. and Narayanan, S.: 2008, The Vera am Mittag German Audio-Visual Emotional Speech Database, *Proceedings of the IEEE International Conference on Multimedia and Expo*.

[75] Groot, P., Birlutiu, A. and Heskes, T.: 2011, Learning from Multiple Annotators with Gaussian Processes, *Proceedings of ICANN*.

[76] Gupta, P. and Rajput, N.: 2007, Two-Stream Emotion Recognition For Call Center Monitoring, *Proceedings of INTERSPEECH*, pp. 2241–2244.

[77] Hardwick, J. and Stout, Q.: 1991, Bandit Strategies for Ethical Sequential Allocation, *Computing Science and Statistics* **23**, 421–424.

[78] Heer, J. and Bostock, M.: 2010, Crowdsourcing Graphical Perception: Using Mechanical Turk to Assess Visualization Design, *Proceedings of CHI*, pp. 203–212.

[79] Ho, C., Jabbari, S. and Vaughan, J.: 2013, Adaptive Task Assignment for Crowdsourced Classification, *Proceedings of ICML*.

[80] Honda, J. and Takemura, A.: 2010, An Asymptotically Optimal Bandit Algorithm for Bounded Support Models, *Proceedings of COLT*.

[81] Hoque, E., Yeasin, M. and Louwerse, M.: 2006, Robust Recognition of Emotion from Speech, *LNAI, IVA 2006* **4133**, 42–53.

[82] Howe, J.: 2008, *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business*, Crown Business.

[83] Hozjan, V. and Kačič, Z.: 2006, A Rule-based Emotion-dependent Feature Extraction Method for Emotion Analysis from Speech, *Journal of the Acoustical Society of America* **119**(5), 3109–3120.

[84] Hsueh, P., Melville, P. and Sindhwani, V.: 2009, Data Quality from Crowdsourcing: A Study of Annotation Selection Criteria, *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pp. 27–35.

[85] Hu, R., Mac Namee, B. and Delany, S.: 2010, Off to a Good Start: Using Clustering to Select the Initial Training Set in Active Learning, *Proceedings of Florida Artificial Intelligence Research Society Conference*.

[86] Hyun, K., Kim, E. and Kwak, Y.: 2005, Improvement of Emotion Recognition by Bayesian Classifier Using Non-zero-pitch Concept, *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication*, pp. 312–316.

[87] Ipeirotis, P., Provost, F. and Wang, J.: 2010, Quality Management on Amazon Mechanical Turk, *Proceedings of the ACM SIGKDD Workshop on Human Computation*.

[88] Iriondo, I., Planet, S., Alías, F., Socoró, J. and Martínez, E.: 2007, Validation of an Expressive Speech Corpus by Mapping Automatic Classification to Subjective Evaluation, *LNCS, IWANN 2007* **4507**, 646–653.

[89] Iriondo, I., Planet, S., Socoró, J. and Alías, F.: 2007, Objective and Subjective Evaluation of an Expressive Speech Corpus, *Proceedings of the International Conference on Advances in Nonlinear Speech Processing*.

[90] Jung, H. J. and Lease, M.: 2012, Improving Quality of Crowdsourced Labels via Probabilistic Matrix Factorization, *Proceedings of the 4th Human Computation Workshop, in conjunction with AAAI*, pp. 101–106.

[91] Kajino, H., Tsuboi, Y. and Kashima, H.: 2012, A Convex Formulation for Learning from Crowds, *Proceedings of AAAI*.

[92] Kajino, H., Tsuboi, Y. and Kashima, H.: 2013, Clustering Crowds, *Proceedings of AAAI*.

[93] Kamar, E., Hacker, S. and Horvitz, E.: 2012, Combining Human and Machine Intelligence in Large-scale Crowdsourcing, *Proceedings of AAMAS*.

[94] Karger, D. R., Oh, S. and Shah, D.: 2013, Efficient Crowdsourcing for Multi-class Labeling, *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems*.

[95] Khattak, F. and Salleb-Aouissi, A.: 2011, Quality Control of Crowd Labeling through Expert Evaluation, *Proceedings of the Second Workshop on Computational Social Science and the Wisdom of Crowds, in conjunction with NIPS*.

[96] Kittur, A., Chi, E. and Suh, B.: 2008, Crowdsourcing for Usability: Using Micro-task Markets for Rapid, Remote, and Low-cost User Measurements, *Proceedings of CHI* .

[97] Know, M. and Mirghafori, N.: 2007, Automatic Laughter Detection Using Neural Networks, *Proceedings of INTERSPEECH*, pp. 2973–2976.

[98] Komatani, K., Ito, R., Kawahara, T. and Okuno, H.: 2004, Recognition of Emotional States in Spoken Dialogue with a Robot, *LNAI, IEA/AIE 2004* **3029**, 413–423.

[99] Kostoulas, T., Mporas, I., Ganchev, T. and Fakotakis, N.: 2008, The Effect of Emotional Speech on a Smart-home Application, *LNAI, IEA/AIE 2008* **5027**, 305–310.

[100] Krajewski, J. and Kroger, B.: 2007, Using Prosodic and Spectral Characteristics for Sleepiness Detection, *Proceedings of INTERSPEECH 2007*, pp. 1841–1844.

[101] Krippendorff, K.: 2004, *Content Analysis: an Introduction to its Methodology*, 2nd edn, Sage.

[102] Kustner, D., Tato, R., Kemp, T. and Meffert, B.: 2004, Towards Real Life Applications in Emotion Recognition, *LNAI, ADS 2004* **3068**, 25–34.

[103] Laskowski, K. and Burger, S.: 2006, Annotation and Analysis of Emotionally Relevant Behavior in the ISL Meeting Corpus, *Proceedings of LREC*.

[104] Law, E.: 2011, Defining (Human) Computation, *Proceedings of the Workshop on Crowdsourcing and Human Computation, in conjunction with CHI*.

[105] Law, E. and Von Ahn, L.: 2011, *Human Computation*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan&Claypool.

[106] Lee, C., Mower, E., Busso, C., Lee, S. and Narayanan, S.: 2009, Emotion Recognition using a Hierarchical Binary Decision Tree Approach, *Proceedings of INTERSPEECH*.

[107] Lee, C. and Narayanan, S.: 2005, Toward Detecting Emotions in Spoken Dialogs, *IEEE Transactions on Speech and Audio Processing* **13**(2), 293–303.

[108] Lee, W., Roh, Y., Kim, D., Kim, J. and Hong, K.: 2008, Speech Emotion Recognition Using Spectral Entropy, *LNAI, ICIRA 2008, Part II* **5315**, 45–54.

[109] Leong, C. and Mihalcea, R.: 2011, Measuring the Semantic Relatedness between Words and Images, *International Conference on Semantic Computing*.

[110] Liscombe, J., Riccardi, G. and Hakkani-Tür, D.: 2005, Using Context to Improve Emotion Detection in Spoken Dialog Systems, *Proceedings of INTERSPEECH*.

[111] Litman, D. and Forbes-Riley, K.: 2006, Recognizing Student Emotions and Attitudes on the Basis of Utterances in Spoken Tutoring Dialogues with both Human and Computer Tutors, *Speech Communication* **48**, 559–590.

[112] Liu, C. and Wang, Y.: 2012, TrueLabel + Confusions: A Spectrum of Probabilistic Models in Analyzing Multiple Ratings, *Proceedings of ICML*.

[113] Liu, Q., Peng, J. and Ihler, A.: 2012, Variational Inference for Crowdsourcing, *Proceedings of NIPS*.

[114] Luce, D.: 1959, *Individual Choice Behavior*, Wiley.

[115] Maffiolo, V., Chateau, N. and Chenadec, G.: 2007, Temporal Organization in Listeners' Perception of the Speakers' Emotions and Characteristics: A Way to Improve the Automatic Recognition of Emotion-Related States in Human Voice, *LNCS, ACII 2007* **4738**, 171–178.

[116] Malone, T., Laubacher, R. and Dellarocas, C.: 2010, The Collective Intelligence Genome, *MIT Sloan Management Review* **51**(3), 21–31.

[117] Mann, H.: 1945, Nonparametric tests against trend, *Econometrica* **13**, 245–259.

[118] Mason, W. and Watts, D.: 2010, Financial Incentives and the "Performance of Crowds", *ACM SIGKDD Explorations Newsletter* **11**, 100–108.

[119] Mauss, I. B. and Robinson, M. D.: 2009, Measures of Emotion: a Review, *Cognition & Emotion* **23**(2), 209–237.

[120] Morrison, D. and De Silva, L.: 2007, Voting Ensembles for Spoken Affect Classification, *Journal of Network and Computer Applications* **30**(4), 1356–1365.

[121] Moschou, V., Ververidis, D. and Kotropoulos, C.: 2006, On the Variants of the Self-Organizing Map That Are Based on Order Statistics, *LNCS, ICANN 2006, Part I* **4131**, 425–434.

[122] Neiberg, D., Elenius, K. and Laskowski, K.: 2006, Emotion Recognition in Spontaneous Speech Using GMMs, *Proceedings of INTERSPEECH*, pp. 809–812.

[123] Nowak, S. and Ruger, S.: 2010, How Reliable are Annotations via Crowdsourcing?, *Proceedings of the International Conference on Multimedia Information Retrieval*, pp. 557–566.

[124] Ortego-Resa, C., Moreno, I., Ramos, D. and Gonzalez-Rodriguez, J.: 2009, Anchor Model Fusion for Emotion Recognition in Speech, *LNCS, Biometric ID Management and Multimodal Communication 2009* **5707**, 49–56.

[125] Pao, T., Chen, Y. and Yeh, J.: 2004, Emotion Recognition from Mandarin Speech Signals, *Proceedings of the International Symposium on Chinese Spoken Language Processing*, pp. 301–304.

[126] Pao, T., Chen, Y., Yeh, J., Cheng, Y. and Lin, Y.: 2007, A Comparative Study of Different Weighting Schemes on KNN-Based Emotion Recognition in Mandarin Speech, *LNCS, ICIC 2007* **4681**, 997–1005.

[127] Paolacci, G., Chandler, J. and Ipeirotis, P.: 2010, Running Experiments on Amazon Mechanical Turk, *Judgment and Decision Making* **5**(5), 411–419.

[128] Park, C. and Sim, K.: 2006, The Novel Feature Selection Method Based on Emotion Recognition System, *LNBI, ICIC 2006* **4115**, 731–740.

[129] Pfeiffer, T., Gao, X., Mao, A., Chen, Y. and Rand, D.: 2012, Adaptive Polling for Information Aggregation, *Proceedings of AAAI*.

[130] Piller, F., Ihl, C. and Vossen, A.: 2010, A Typology of Customer Co-creation in the Innovation Process, http://ssrn.com/abstract=1732127.

[131] Quafafou, C. W. and Mohamed: 2012, Learning from Multiple Naive Annotators, *LNAI* **7713**, 173–185.

[132] Quinn, A. and Bederson, B.: 2009, A Taxonomy of Distributed Human Computation, *Technical report*.

[133] Quinn, A. and Bederson, B.: 2011, Human Computation: a Survey and Taxonomy of a Growing Field, *Proceedings of CHI*.

[134] Raykar, V. C. and Yu, S.: 2012, Eliminating Spammers and Ranking Annotators for Crowdsourced Labeling Tasks, *JMLR* **13**, 491–518.

[135] Raykar, V., Yu, S., Zhao, L., Jerebko, A., Florin, C., Valadez, G., Bogoni, L. and Moy, L.: 2009, Supervised Learning from Multiple Experts: whom to Trust when Everyone Lies a Bit, *Proceedings of ICML*.

[136] Raykar, V., Yu, S., Zhao, L., Valadez, G., Florin, C., Bogoni, L. and Moy, L.: 2010, Learning From Crowds, *Journal of Machine Learning Research* **11**, 1297–1322.

[137] Razak, A., Komiya, R., Izani, M. and Abidin, M.: 2005, Comparison between Fuzzy and NN Method for Speech Emotion Recognition, *Proceedings of the Third International Conference on Information Technology and Applications*, pp. 297–302.

[138] Reynolds, D., Quatieri, T. and Dunn, R.: 2000, Speaker Verification Using Adapted Gaussian Mixture Models, *Digital Signal Processing* **10**(1-3), 19–41.

[139] Robbins, H.: 1952, Some Aspects of the Sequential Design of Experiments, *Bulletin of the American Mathematics Society* **58**, 527–535.

[140] Rodrigues, F., Pereira, F. and Ribeiro, B.: 2013, Learning from Multiple Annotators: Distinguishing Good from Random Labelers, *Pattern Recognition Letters* **34**(12), 1428–1436.

[141] Rouse, A.: 2010, A Preliminary Taxonomy of Crowdsourcing, *Proceedings of the 21st Australasian Conference on Information Systems*.

[142] Schenk, E. and Guittard, C.: 2011, Towards a Characterization of Crowdsourcing Practices, *Journal of Innovation Economics* **7**(1), 93–107.

[143] Scherer, K. and Ekman, P. (eds): 1984, *Approaches to Emotion*.

[144] Scherer, S., Oubbati, M., Schwenker, F. and Palm, G.: 2008, Real-Time Emotion Recognition from Speech Using Echo State Networks, *LNAI, ANNPR 2008* **5064**, 205–216.

[145] Schroder, M.: 2004, Dimensional Emotion Representation as a Basis for Speech Synthesis with Non-extreme Emotions, *LNCS, Proceedings of the Kloster Irsee Tutorial and Research Workshop on Affective Dialogue Systems* **3068**, 209–220.

[146] Schuller, B., Arsic, D., Wallhoff, F. and Rigoll, G.: 2006, Emotion Recognition in the Noise Applying Large Acoustic Feature Sets, *Proceedings of Speech Prosody*, pp. 276–289.

[147] Schuller, B., Batliner, A., Seppi, D., Steidl, S., Vogt, T., Wagner, J., Devillers, L., Vidrascu, L., Amir, N., Kessous, L. and Aharonson, V.: 2007, The Relevance of Feature Type for the Automatic Classification of Emotional User States: Low Level Descriptors and Functionals, *Proceedings of INTERSPEECH*.

[148] Schuller, B., Muller, R., Lang, M. and Rigoll, G.: 2005, Speaker Independent Emotion Recognition by Early Fusion of Acoustic and Linguistic Features within Ensembles, *Proceedings of INTERSPEECH*, pp. 805–808.

[149] Schuller, B., Reiter, S., Muller, R., Al-Hames, M., Lang, M. and Rigoll, G.: 2005, Speaker Independent Speech Emotion Recognition by Ensemble Classification, *Proceedings of the IEEE International Conference on Multimedia and Expo*, pp. 864–867.

[150] Schuller, B., Reiter, S. and Rigoll, G.: 2006, Evolutionary Feature Generation in Speech Emotion Recognition, *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 5–8.

[151] Schuller, B., Rigoll, G. and Lang, M.: 2003, Hidden Markov Model-based Speech Emotion Recognition, *Proceedings of ICASSP*.

[152] Schuller, B., Seppi, D., Batliner, A., Maier, A. and Steidl, S.: 2007, Towards More Reality in the Recognition of Emotional Speech, *Proceedings of ICASSP*, pp. 941–944.

[153] Schuller, B., Steidl, S. and Batliner, A.: 2009, The INTERSPEECH 2009 Emotion Challenge, *Proceedings of INTERSPEECH* .

[154] Schultz, R., Peter, C., Blech, M., Voskamp, J. and Urban, B.: 2007, Towards Detecting Cognitive Load and Emotions in Usability Studies Using the RealEYES Framework, *LNCS, NCII 2007* **4559**, 412–421.

[155] Schwenker, F., Scherer, S., Magdi, Y. and Palm, G.: 2009, The GMM-SVM Supervector Approach for the Recognition of the Emotional Status from Speech, *LNCS, ICANN 2009, Part 1* **5768**, 894–903.

[156] Scott, S.: 2010, A Modern Bayesian Look at the Multi-armed Bandit, *Applied Stochastic Models in Business and Industry* **26**, 639–658.

[157] Seppi, D., Batliner, A., Schuller, B., Steidl, S., Vogt, T., Wagner, J., Devillers, L., Vidrascu, L., Amir, N. and Aharonson, V.: 2008, Patterns, Prototypes, Performance Classifying Emotional User States, *Proceedings of INTERSPEECH* pp. 601–604.

[158] Settles, B.: 2010, Active Learning Literature Survey, *Technical report*.

[159] Shafran, I., Riley, M. and Mohri, M.: 2003, Voice Signatures, *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 31–36.

[160] Shami, M. and Verhelst, W.: 2007, Automatic Classification of Expressiveness in Speech: A Multi-corpus Study, *LNAI, Speaker Classification II* **4441**, 43–56.

[161] Singla, A. and Krause, A.: 2013, Truthful Incentives in Crowdsourcing Tasks using Regret Minimization Mechanisms, *Proceedings of WWW*.

[162] Slaney, M. and McRoberts, G.: 2003, Baby Ears: A Recognition System for Affective Vocalizations, *Speech Communication* **39**(3-4), 367–384.

[163] Smyth, P., Fayyad, U., Burl, M., Perona, P. and Baldi, P.: 1995, Inferring Ground Truth from Subjective Labelling of Venus Images, *Proceedings of NIPS*, pp. 1085–1092.

[164] Snel, J., Tarasov, A., Cullen, C. and Delany, S.: 2012, A Crowdsourcing Approach to Labelling a Mood Induced Speech Corpus, *Proceedings of the 4th International Workshop on Corpora for Research on Emotion Sentiment & Social Signals, in conjunction with LREC*.

[165] Snow, R., O'Connor, B., Jurafsky, D. and Ng, A. Y.: 2008, Cheap and Fast—But Is It Good? Evaluating Non-expert Annotations for Natural Language Tasks, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[166] Soleymani, M. and Larson, M.: 2010, Crowdsourcing for Affective Annotation of Video: Development of a Viewer-reported Boredom Corpus, *Proceedings of the Workshop on Crowdsourcing for Search Evaluation, in conjunction with SIGIR*.

[167] Soliman, M. A., Ilyas, I. F., Martinenghi, D. and Tagliasacchi, M.: 2011, Ranking with uncertain scoring functions: Semantics and sensitivity measures, *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pp. 805–816.

[168] Sorokin, A. and Forsyth, D.: 2008, Utility Data Annotation with Amazon Mechanical Turk, *Proceedings of CVPR*.

[169] Staroniewicz, P.: 2009, Recognition of Emotional State in Polish Speech—Comparison between Human and Automatic Efficiency, *LNCS, Biometric ID Management and Multimodal Communication 2009* **5707**, 33–40.

[170] Steidl, S.: 2009, *Automatic Classification of Emotion-related User States in Spontaneous Children's Speech*, PhD thesis, Erlangen-Nurnberg University.

[171] Steidl, S., Levit, M., Batliner, A., Nöth, E. and Niemann, H.: 2005, "Of All Things the Measure Is Man": Automatic Classification of Emotions and Inter-labeler Consistency, *Proceedings of ICASSP*, pp. 317–320.

[172] Su, Q., Pavlov, D., Chow, J. and Baker, W.: 2007, Internet-scale Collection of Human-reviewed Data, *Proceedings of WWW*.

[173] Tarasov, A., Cullen, C. and Delany, S.: 2010, Using Crowdsourcing for Labelling Emotional Speech Assets, *Proceedings of W3C Workshop on Emotion Markup Language*.

[174] Tarasov, A. and Delany, S.: 2011, Benchmarking Classification Models for Emotion Recognition in Natural Speech: A Multi-corporal Study, *Proceedings of IEEE International Conference on Automatic Face & Gesture Recognition and Workshops*, pp. 841–846.

[175] Tarasov, A., Delany, S. and Mac Namee, B.: 2012a, Dynamic Estimation of Rater Reliability in Regression Tasks using Multi-Armed Bandit Techniques, *Proceedings of Workshop on Machine Learning in Human Computation and Crowdsourcing, in conjunction with ICML*.

[176] Tarasov, A., Delany, S. and Mac Namee, B.: 2012b, Dynamic Estimation of Rater Reliability in Subjective Tasks Using Multi-Armed Bandits, *Proceedings of ASE/IEEE International Conference on Social Computing*.

[177] Tarasov, A., Delany, S. and Mac Namee, B.: 2013, Improving Performance by Re-Rating in the Dynamic Estimation of Rater Reliability, *Proceedings of Machine Learning Meets Crowdsourcing Workshop, in conjunction with ICML*.

[178] Tarasov, A., Delany, S. and Mac Namee, B.: 2014, Dynamic estimation of worker reliability in crowdsourcing for regression tasks: Making it work, *Expert Systems with Applications* **41**, 6190–6210.

[179] Thiel, C., Scherer, S. and Schwenker, F.: 2007, Fuzzy-Input Fuzzy-Output One-Against-All Support Vector Machines, *LNAI, KES 2007/WIRN 2007, Part III* **4694**, 156–165.

[180] Toth, S., Sztaho, D. and Vicsi, K.: 2008, Speech Emotion Perception by Human and Machine, *LNAI, HH and HM Interaction 2007* **5042**, 213–224.

[181] Tran-Thanh, L., Stein, S., Rogers, A. and Jennings, N.: 2012, Efficient Crowdsourcing of Unknown Experts using Multi-Armed Bandits, *Proceedings of ECAI*.

[182] Triantaphyllou, E. and Baig, K.: 2005, The Impact of Aggregating Benefit and Cost Criteria in Four MCDA Methods, *IEEE Transactions on Engineering Management* **52**(2), 213–226.

[183] Truong, K. and Leeuwen, D.: 2007, Visualizing Acoustic Similarities between Emotions in Speech: An Acoustic Map of Emotions, *Proceedings of INTERSPEECH*, pp. 2265–2268.

[184] Truong, K. and Raaijmakers, S.: 2008, Automatic Recognition of Spontaneous Emotions in Speech Using Acoustic and Lexical Features, *LNCS, MLMI 2008* **5237**, 161–172.

[185] Valizadegan, H., Nguyen, Q. and Hauskrecht, M.: 2012, Learning Medical Diagnosis Models from Multiple Experts, *AMIA Annual Symposium Proceedings* pp. 921–930.

[186] Vaughan, B.: 2011, *Naturalistic Emotional Speech Corpora with Large Scale Emotional Dimension Ratings*, PhD thesis, Dublin Institute of Technology.

[187] Vermorel, J. and Mohri, M.: 2005, Multi-armed Bandit Algorithms and Empirical Evaluation, *LNAI, Machine Learning: ECML 2005* **3720**, 437–448.

[188] Ververidis, D. and Kotropoulos, C.: 2006, Emotional Speech Recognition: Resources, Features, and Methods, *Speech Communication* **48**(9), 1162–1181.

[189] Vidrascu, L. and Devillers, L.: 2005, Real-life Emotion Representation and Detection in Call Centers Data, *Proceedings of the First International Conference on Affective Computing and Intelligent Interaction*.

[190] Vidrascu, L. and Devillers, L.: 2007, Five Emotion Classes Detection in Real-world Call Center Data: The Use of Various Types of Paralinguistic Features, *Proceedings of the International Workshop on Paralinguistic Speech*, pp. 11–16.

[191] Vlasenko, B., Schuller, B., Wendemuth, A. and Rigoll, G.: 2007, Frame vs. Turn-level: Emotion Recognition from Speech Considering Static and Dynamic Processing, *LNCS, ACII 2007* **4738**, 139–147.

[192] Vogt, T., Andre, E. and Bee, N.: 2008, EmoVoice—A Framework for Online Recognition of Emotions from Voice, *Proceedings of the 4th IEEE Tutorial and Research*

Workshop on Perception and Interactive Technologies for Speech-Based Systems: Perception in Multimodal Dialogue Systems.

[193] Vondra, M. and Vich, R.: 2009, Evaluation of Speech Emotion Classification Based on GMM and Data Fusion, *LNAI, Cross-Modal Analysis* **5641**, 98–105.

[194] Wagner, J., Kim, J. and Andre, E.: 2005, From Physiological Signals to Emotions: Implementing and Comparing Selected Methods for Feature Extraction and Classification, *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 940–943.

[195] Wallace, B., Small, K. and Brodley, C.: 2011, Who Should Label What? Instance Allocation in Multiple Expert Active Learning, *Proceedings of the SIAM International Conference on Data Mining.*

[196] Welinder, P., Branson, S., Perona, P. and Belongie, S. J.: 2010, The Multidimensional Wisdom of Crowds, *Proceedings of NIPS*, pp. 2424–2432.

[197] Welinder, P. and Perona, P.: 2010, Online Crowdsourcing: Rating Annotators and Obtaining Cost-effective Labels, *Workshop on Advancing Computer Vision with Humans in the Loop, in conjunction with CVPR.*

[198] Whitehill, J., Ruvolo, P., Wu, T., Bergsma, J. and Movellan, J.: 2009, Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise, *Proceedings of NIPS*, pp. 2035–2043.

[199] Whitla, P.: 2009, Crowdsourcing and its Application in Marketing Activities, *Contemporary Management Research* **5**(1), 15–28.

[200] Wightman, D.: 2010, Crowdsourcing Human-Based Computation, *Proceedings of NordiCHI '10*, pp. 551–560.

[201] Wu, W., Liu, Y., Guo, M., Wang, C. and Liu, X.: 2013, A Probabilistic Model of Active Learning with Multiple Noisy Oracles, *Neurocomputing* **118**(C), 253–262.

[202] Xiang Liu, L. L. and Memon, N.: 2013, A Lightweight Combinatorial Approach for Inferring the Ground Truth from Multiple Annotators, *LNAI* pp. 616–628.

[203] Xiao, H., Xiao, H. and Eckert, C.: 2013, Learning from Multiple Observers with Unknown Expertise, *LNCS* **7818**, 595–606.

[204] Xiao, Z., Dellandrea, E., Chen, L. and Dou, W.: 2009, Recognition of Emotions in Speech by a Hierarchical Approach, *Proceedings of the 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops.*

[205] Yan, Y., Rosales, R., Fung, G. and Dy, J.: 2011, Active Learning from Crowds, *Proceedings of ICML.*

[206] Ye, C., Liu, J., Chen, C., Song, M. and Bu, J.: 2008, Speech Emotion Classification on a Riemannian Manifold, *Proceedings of Advances in Multimedia Information Processing*, pp. 61–69.

[207] Yilmazyildiz, S., Mattheyses, W., Patsis, Y. and Verhelst, W.: 2006, Expressive Speech Recognition and Synthesis as Enabling Technologies for Affective Robot-Child Communication, *LNCS, PCM 2006* **4261**.

[208] Yoon, W., Cho, Y. and Park, K.: 2007, A Study of Speech Emotion Recognition and Its Application to Mobile Services, *LNCS, UIC 2007* **4611**, 758–766.

[209] Yoon, W. and Park, K.: 2007, A Study of Emotion Recognition and Its Applications, *LNAI, MDAI 2007* **4617**, 455–462.

[210] You, M., Li, G., Chen, L. and Tao, J.: 2008, A Novel Classifier Based on Enhanced Lipschitz Embedding for Speech Emotion Recognition, *LNCS, ICIC 2008* **5226**, 482–490.

[211] Zeicher, N., Berant, J. and Dagan, I.: 2012, Crowdsourcing Inference-Rule Evaluation, *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics.*

[212] Zeng, Z., Hu, Y., Roisman, G., Wen, Z., Fu, Y. and Huang, T.: 2007, Audio-visual Spontaneous Emotion Recognition, *LNCS, Artificial Intelligence for Human Computing* **4451**, 72–90.

[213] Zhang, P. and Obradovic, Z.: 2011, Learning from Inconsistent and Unreliable Annotators by a Gaussian Mixture Model and Bayesian Information Criterion , *LNAI* **6913**, 553–568.

[214] Zhang, S.: 2008, Emotion Recognition in Chinese Natural Speech by Combining Prosody and Voice Quality Features, *LNCS, ISNN 2008, Part II* **5264**, 457–464.

[215] Zhou, D., Platt, J., Basu, S. and Mao, Y.: 2012, Learning from the Wisdom of Crowds by Minimax Entropy, *Proceedings of NIPS*.

[216] Zou, J. and Parkes, D.: 2012, Get Another Worker? Active Crowdlearning with Sequential Arrivals., *Proceedings of the Machine Learning in Human Computation & Crowdsourcing Workshop, in conjunction with ICML*.

[217] Zwass, V.: 2010, Co-Creation: Toward a Taxonomy and an Integrated Research Perspective, *International Journal of Electronic Commerce* **15**(1), 11–48.