

DISS. ETH NO. 24017

Quality Control and Optimization for Hybrid Crowd-Machine Learning Systems

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

BESMIRA NUSHI

Master of Science in Computer Science
RWTH Aachen University, Germany
University of Trento, Italy

born on 12.06.1984

citizen of Albania

accepted on the recommendation of

Prof. Dr. Donald Kossmann (ETH Zurich), examiner
Prof. Dr. Andreas Krause (ETH Zurich), co-examiner
Dr. Eric Horvitz (Microsoft Research), co-examiner

2016

Abstract

Human computation has traditionally been an essential mechanism for providing training data and feedback to machine learning algorithms. Until a decade ago, human input was collected mainly from machine learning experts or via controlled user studies with small groups of people. With the rapid development of Internet technologies, human computation became applicable to problems that require large-scale training data. To this end, crowdsourcing is a form of human computation facilitated by online frameworks on the Internet, which in their simplest model serve as shared marketplaces for both crowd requesters and crowd workers.

This dissertation focuses on two aspects of integrating crowdsourcing in the process of building and improving machine learning algorithms and systems. First, it studies how *human supervision* can be efficiently leveraged for generating training label data for new machine learning models and algorithms. Second, it explores the impact of *human intervention* for assisting machine learning experts in troubleshooting and improving existing systems composed of multiple machine learning components.

While crowdsourcing opens promising opportunities in supporting machine learning techniques, it also poses new challenges relevant to both human supervision and intervention in intelligent systems. As opposed to expert input, crowdsourcing data may involve noise which lowers the quality of the collected data and the corresponding predictions. Noise is present in crowdsourcing data due to possible subjectivity, ambiguous task design, human error, and insufficient qualification worker skills. In order to accommodate *quality control* measures that account for noise, machine learning models need to be appropriately adopted for representing and interpreting crowd data sources. Moreover, due to the large size of datasets and the design space of machine learning models, crowdsourcing supervision and intervention can be costly and often not feasible. For this purpose, *cost optimization* mechanisms are necessary for scaling the crowdsourcing process and making it affordable even for complex tasks that have high data requirements. In order to tackle the two challenges of (possibly) noisy and

costly crowd input, this thesis contributes towards building quality control and cost optimization techniques for hybrid crowd-machine systems that learn and are improved from human-generated data.

The first contribution of the thesis is a crowd model, which we call the Access Path Model. It seamlessly tackles the problems of label aggregation and cost optimization for making new predictions. Differently from what has been proposed in previous work, the Access Path Model relies on group-based representations of the crowd named as access paths. This high-level abstraction allows the model to express worker answer correlations in addition to the worker individual profiles. The design is beneficial for making robust decisions with meaningful confidence even in the presence of noise and sparse worker participation. Moreover, it allows for efficient crowd access optimization schemes, which plan the budget allocation to diverse access paths in order to maximize the information gain for new predictions. Closely related to this contribution, we then investigate cost optimization strategies that can be applied in the early stage of collecting training data for a new model. In this context, we propose the B-LEAFS algorithm, which dynamically acquires data for feature-based classification models. B-LEAFS naturally trades off exploration and exploitation crowd access decisions and overcomes the challenge of data insufficiency via model sampling and parameter credibility checks.

The Access Path Model and the B-LEAFS algorithm are strategies of quality control and cost optimization for building a single machine learning model from crowdsourced labels. In the quest of a deeper integration of human computation with complete intelligent systems, the final contribution of this thesis is a troubleshooting methodology for integrative computational pipelines composed of multiple machine learning components. The goal of the methodology is to guide system designers in the process of decision-making for improving the quality of current systems. For this purpose, the methodology involves human computation for simulating component fixes that cannot be generated otherwise. The simulated fixes are injected back in the system execution, which allows for systematic analysis of the potential impact of individual and joint fixes in the overall system output quality. This human-assisted methodology is a powerful tool for better understanding complex systems and prioritizing research and engineering efforts towards future system enhancements.

Zusammenfassung

Human Computation ist ein wichtiger Mechanismus zur Bereitstellung von Trainingsdaten und Feedback für maschinelles Lernen. Bis vor einem Jahrzehnt wurde dieser menschliche Aspekt hauptsächlich manuell durch Experten oder kontrollierte Benutzerstudien mit einer kleinen Zahl von Teilnehmern verwirklicht. Mit der rasanten Entwicklung des Internets ist Human Computation zunehmend für Probleme mit Trainingsdaten in großem Umfang anwendbar. Crowdsourcing ist ein Beispiel dafür: Hier wird Human Computation durch ein Online-Framework verwirklicht, das in seiner einfachsten Form als Marktplatz sowohl für Anbieter als auch Crowd-Arbeiter dient.

Diese Doktorarbeit legt ihren Schwerpunkt auf die Integration von Crowdsourcing für den Entwurf und die Verbesserung von Systemen und Algorithmen im Bereich des maschinellen Lernens. Zunächst wird untersucht wie menschliche Überwachung effizient genutzt werden kann um die Klassifizierung von Trainingsdaten für neue Modelle und Algorithmen für maschinelles Lernen zu erzeugen. Dann wird der Einfluss menschlicher Eingriffe zur Unterstützung von Experten bei der Behebung und Verbesserung von existierenden Systemen, die aus mehreren Komponenten des maschinellen Lernens zusammengebaut sind, untersucht.

Crowdsourcing bietet attraktive neue Möglichkeiten zur Unterstützung existierender Techniken im maschinellen Lernen, aber birgt auch neue Herausforderungen bezüglich menschlicher Überwachung und menschlicher Eingriffe in intelligente Systeme. Im Gegensatz zum Beitrag von Experten sind Crowdsourcing-Daten oft ungenau. Dies senkt die Qualität der gesammelten Daten und der dazugehörigen Vorhersagen. Die vorhandenen Ungenauigkeiten in den Daten sind oft ein Resultat von Subjektivität, mehrdeutigem Design der Tasks, menschlichen Fehlern oder der ungenügender Qualifikation von Arbeitern. Um eine Qualitätskontrolle zur Verfügung zu stellen, die mit verzerrten Daten umgehen kann, müssen Modelle des maschinellen Lernens für die Repräsentation und Interpretation entsprechend angepasst werden. Als Konsequenz der Größe der Daten und des Design-Spielraums von Modellen sind die Aufsicht und

der Eingriff durch Crowdsourcing oft teuer und deshalb nicht realistisch.

Aus diesem Grund sind Mechanismen zur Kostenoptimierung nötig um Crowdsourcing-Prozesse zu skalieren und bezahlbar zu machen - insbesondere auch, für komplexe Aufgaben mit hohen Anforderungen an die Daten. Diese Doktorarbeit leistet einen Beitrag zur Kostenreduzierung und Minderung der Störanfälligkeit beim Entwurf von Techniken zur Qualitätskontrolle und Kostenoptimierung für hybride Crowdsourcing-Systeme, die lernen und durch menschlich generierte Daten verbessert werden.

Der erste Beitrag dieser Ausarbeitung ist ein Modell des Crowdsourcing, welches wir Access Path Modell nennen. Es löst nahtlos Probleme bei der Aggregation und Kostenoptimierung zur Erzeugung von Vorhersagen. Im Unterschied zu existierenden Arbeiten basiert das Access Path Modell auf gruppen-basierten Repräsentationen der Crowd, die wir Access Path nennen. Diese Abstraktion bindet zusätzlich zu individuellen Arbeiter-Profilen die Korrelation von Antworten zu Arbeitern in das Modell ein. Dieses Design ist hilfreich um robuste Entscheidungen mit nützlicher Zuverlässigkeit selbst dann zu gewähren, wenn Störungen vorliegen oder nur wenige Arbeiter teilgenommen haben. Weiter erlaubt das Design die Verteilung des Budgets auf unterschiedliche Access Paths und ermöglicht somit eine effiziente Optimierung des Zugangs zur Crowd. Dies maximiert den Informationsgewinn. Im Anschluss daran werden Strategien zur Kostenoptimierung untersucht, die in den frühen Phasen des Sammelns von Trainingsdaten für neue Modelle angewendet werden können. In diesem Kontext stellen wir unseren B-LEAFS Algorithmus vor, welcher dynamisch Daten für merkmalsbasierte Klassifikationsmodelle erarbeitet. B-LEAFS bietet einen Kompromiss zwischen der Erkundung und Erschließung des Zugang zur Crowd und löst das Problem von unzureichenden Daten nach dem Sampling des Modells und bei der Überprüfung der Glaubwürdigkeit von Parametern.

Auf der Suche nach tiefgreifender Integration von Human Computation in abgeschlossenen Intelligenten Systemen bietet diese Doktorarbeit als letzten Beitrag eine Methode zur Fehlerbehandlung von Komponenten in intelligenten Systemen. Das Ziel dieser Methode ist den Designer des Systems bei der Verbesserung der Qualität bestehender Systeme zu unterstützen. Dafür verwenden wir Human Computation zur Simulation der Fehlerbehebung von Komponenten, die anderweitig nicht erzeugt werden können. Die simulierte Fehlerbehebung wird dann bei der Ausführung der Systeme eingefügt. Dies ermöglicht die systematische Analyse des Einflusses von individuellen und kombinierten Fehlerkorrekturen auf die Qualität der Ausgabe des Gesamtsystems. Diese von Menschen unterstützte Methode ist ein mächtiges Werkzeug zum besseren Verständnis von komplexen Systemen und der Priorisierung von Bemühungen zur Forschung und Entwicklung in Richtung zukünftiger Verbesserungen der Systeme.

Acknowledgments

First, I would like to thank my advisor Donald Kossmann. Donald not only contributed with insightful ideas but he was also a role model with a strong sense of critical thinking which helped me to continuously improve my work and become a better researcher. I thank him for giving me the freedom to explore new directions and following me in this path. Also, I am heavily indebted to my co-advisor Andreas Krause for his trust and support during my doctoral studies. I thank Andreas for making me part of his group, for his patient guidance while dealing with hard problems, and most importantly for inspiring me to pursue my interests in machine learning.

Moreover, I am grateful to Ece Kamar and Eric Horvitz at Microsoft Research for being great mentors during my internship and for their ongoing collaboration afterwards. Ece has always been a bright motivator and mentor of our work in troubleshooting intelligent systems. I thank Eric for his sharp insights on the vision of the work, his constructive comments on the thesis, and for coming in person for my defense.

I am thankful to my long-lasting collaborators at ETH, Adish Singla and Anja Grünheid, who played an important role in our joint work for quality control and optimization in crowdsourcing and made conference traveling more enjoyable. My thanks extend to Erfan Zamanian, Lynn Aders, and Matteo Pozzetti, who worked with me during their Master Thesis and revealed interesting aspects relevant to this thesis.

Next, I would like to thank Eva, Jena, Karel, Nadia, and Simonetta for being friendly and helpful and making the Systems Group a pleasant workplace. My very special thanks go to Anja, Claude, Darko, Gerd, Ingo, Jana, Lukas, Pratanu, Pravin, and Stefan, for being both colleagues and friends. I am happy to have met them all, and to have shared with them my travels, ski trips, dinners, coffee breaks, and spontaneous daily encounters that made my time at ETH a wonderful life experience.

I am forever grateful to my family for motivating and supporting me along my studies from the very first steps to this day. Finally, I thank Marco for being a life partner, a friend, and the first one to listen to my ideas and read my work.

Contents

| | | |
|----------|----------------------------------------------------------|-----------|
| 1 | Introduction | 1 |
| 1.1 | Motivation | 1 |
| 1.1.1 | Human supervision of machine learning models | 3 |
| 1.1.2 | Human intervention on machine learning systems | 5 |
| 1.2 | Background | 6 |
| 1.3 | Contribution | 8 |
| 1.4 | Thesis overview | 10 |
| 2 | Label Aggregation in Crowdsourcing | 13 |
| 2.1 | Overview | 13 |
| 2.2 | Related work | 16 |
| 2.3 | Problem definition: Label Aggregation | 18 |
| 2.4 | Alternative aggregation models | 20 |
| 2.4.1 | Majority-based aggregation | 20 |
| 2.4.2 | Individual models | 20 |
| 2.5 | Access path based models | 22 |
| 2.5.1 | Access Path design | 23 |
| 2.5.1.1 | Architectural context and implications | 23 |
| 2.5.1.2 | Access path configuration and use cases | 24 |
| 2.5.1.3 | Access path discovery | 24 |
| 2.5.2 | Access Path based models | 26 |

Contents

| | | |
|----------|-----------------------------------------------|-----------|
| 2.5.2.1 | Naïve Bayes for Access Paths (NBAP) | 26 |
| 2.5.2.2 | Access Path Model overview | 26 |
| 2.5.2.3 | Parameter learning | 28 |
| 2.5.2.4 | Inference | 30 |
| 2.6 | Experimental evaluation | 31 |
| 2.6.1 | Metrics | 31 |
| 2.6.2 | Dataset description | 32 |
| 2.6.3 | Model evaluation on real-world predictions | 34 |
| 2.6.3.1 | Predictions with unconstrained budget | 35 |
| 2.6.3.2 | Predictions with constrained budget | 36 |
| 2.6.4 | Label noise impact on predictions | 37 |
| 2.7 | Summary | 40 |
| 3 | Crowd Access Optimization | 41 |
| 3.1 | Overview | 41 |
| 3.2 | Related work | 42 |
| 3.3 | Problem definition: Access path selection | 44 |
| 3.4 | Information Gain as a quality measure | 46 |
| 3.4.1 | Information gain computation | 47 |
| 3.4.2 | Submodularity of information gain | 48 |
| 3.5 | Access path selection | 49 |
| 3.5.1 | Greedy approximation scheme | 49 |
| 3.5.2 | Greedy approximation with varying costs | 50 |
| 3.6 | Experimental Evaluation | 51 |
| 3.6.1 | Greedy approximation and diversity management | 51 |
| 3.6.2 | Optimization with budget constraints | 53 |
| 3.6.3 | Impact of diversity | 55 |
| 3.7 | Summary | 57 |

| | | |
|--------------|---------------------------------------------------------------------------------------|---------------|
| 4 | Budgeted Learning and Feature Selection | 59 |
| 4.1 | Overview | 59 |
| 4.2 | Related work | 62 |
| 4.3 | Problem definition: Learning and feature selection under budget constraints | 63 |
| 4.3.1 | Feature-based classification models | 64 |
| 4.3.2 | Problem statement | 65 |
| 4.4 | Budgeted Learning and Feature Selection | 67 |
| 4.4.1 | Existing approaches | 67 |
| 4.4.2 | Limitations and challenges | 68 |
| 4.4.3 | B-LEAFS for Naïve Bayes | 68 |
| 4.4.3.1 | Model sampling | 70 |
| 4.4.3.2 | Submodular feature selection | 70 |
| 4.4.3.3 | Parameter credibility check | 71 |
| 4.4.3.4 | Discussion | 71 |
| 4.4.4 | B-LEAFS for the Access Path model | 73 |
| 4.4.4.1 | Model sampling | 73 |
| 4.4.4.2 | Parameter credibility check | 74 |
| 4.5 | Experimental evaluation | 74 |
| 4.5.1 | Baselines | 74 |
| 4.5.2 | Evaluation on the Naïve Bayes model | 78 |
| 4.5.3 | Evaluation on the Access Path model | 80 |
| 4.5.4 | Noise impact | 81 |
| 4.6 | Summary | 84 |
| 5 | Troubleshooting Machine Learning Systems via Crowdsourcing | 85 |
| 5.1 | Overview | 85 |
| 5.2 | Related work | 87 |
| 5.3 | Case study: An image captioning system | 89 |
| 5.3.1 | System architecture | 89 |

Contents

| | | |
|---------|-------------------------------------------------------|-----|
| 5.3.2 | MSCOCO dataset | 91 |
| 5.4 | Problem characterization | 91 |
| 5.4.1 | Problem context | 92 |
| 5.4.1.1 | System architecture definition | 92 |
| 5.4.1.2 | Quality definition | 92 |
| 5.4.2 | Problem definition | 93 |
| 5.4.3 | Problem characteristics | 93 |
| 5.4.3.1 | Continuous quality measures. | 93 |
| 5.4.3.2 | Complex component entanglement | 94 |
| 5.4.3.3 | Non-monotonic error | 95 |
| 5.5 | Human-in-the-loop methodology | 96 |
| 5.5.1 | Human computation fixes | 96 |
| 5.5.2 | Methodology setup | 97 |
| 5.5.3 | Troubleshooting steps | 97 |
| 5.5.4 | Troubleshooting outcomes | 98 |
| 5.6 | Troubleshooting the image captioning system | 98 |
| 5.6.1 | System evaluation | 99 |
| 5.6.2 | Component fixes | 99 |
| 5.6.2.1 | Visual Detector fixes | 100 |
| 5.6.2.2 | Language Model fixes | 101 |
| 5.6.2.3 | Caption Reranker fixes. | 101 |
| 5.7 | Experimental evaluation | 102 |
| 5.7.1 | Quality scores | 102 |
| 5.7.2 | How does the system fail? | 103 |
| 5.7.2.1 | The current system state | 103 |
| 5.7.2.2 | The current component state | 104 |
| 5.7.3 | Component fixes | 105 |
| 5.7.3.1 | Visual Detector fixes | 105 |
| 5.7.3.2 | Language Model | 107 |

| | | |
|----------|----------------------------------------------------------|------------|
| 5.7.3.3 | Caption Reranker fixes | 107 |
| 5.7.3.4 | Complete fix workflow | 108 |
| 5.7.4 | Quality control | 110 |
| 5.7.5 | Methodology cost | 111 |
| 5.7.6 | Examples of fix integration | 112 |
| 5.8 | How to improve the system? | 113 |
| 5.9 | Discussion: Use cases and generalizability | 114 |
| 5.10 | Summary | 115 |
| 6 | Conclusions and future work | 117 |
| 6.1 | Summary and conclusions | 117 |
| 6.1.1 | Human supervision of machine learning models | 117 |
| 6.1.2 | Human intervention in machine learning systems | 118 |
| 6.2 | Future work | 119 |
| 6.2.1 | Crowdsourcing platform support and integration | 119 |
| 6.2.2 | Crowdsourcing in future intelligent systems | 121 |
| | Appendices | 125 |
| A | Access path selection | 127 |
| A.1 | Proof of Theorem 1 | 127 |
| A.2 | Proof of Theorem 2 | 131 |

1

Introduction

1.1 Motivation

Rapid developments in artificial intelligence and machine learning have given rise to intelligent systems which are now ubiquitous in industry and in our everyday life. Prominent examples include applications of AI in healthcare, transportation, personal assistants as well as entertainment. Despite the improved capabilities of the various algorithms and systems, there is still a large spectrum of tasks that machines are not yet able to handle. Therefore, most of the applications in use nowadays require human input in two main aspects: (i) *Human supervision* — acquiring label data for training the underlying learning algorithms, and (ii) *Human intervention* — recovering from potential errors occurring in the system. The first aspect is related to the process of collecting and aggregating label data from humans for the purpose of training supervised machine learning models. Due to the increasing need for label data, this line of research has attained significant attention from interdisciplinary studies in the intersection of human computation and machine learning [70, 33, 164]. The latter aspect concerns the opportunities of leveraging human intelligence to either bridge or correct shortcomings of intelligent systems. Such human interventions are often crucial given that current systems are not fully autonomous and they make mistakes [132, 111], which requires ongoing human control to ensure reliability especially in mission-critical applications. Both directions motivate the emerging need for building *hybrid human and machine learning systems* where people either supervise or complement machine capabilities to solve complex challenges [69, 83]. Figure 1.1 depicts a high-level view of such systems.

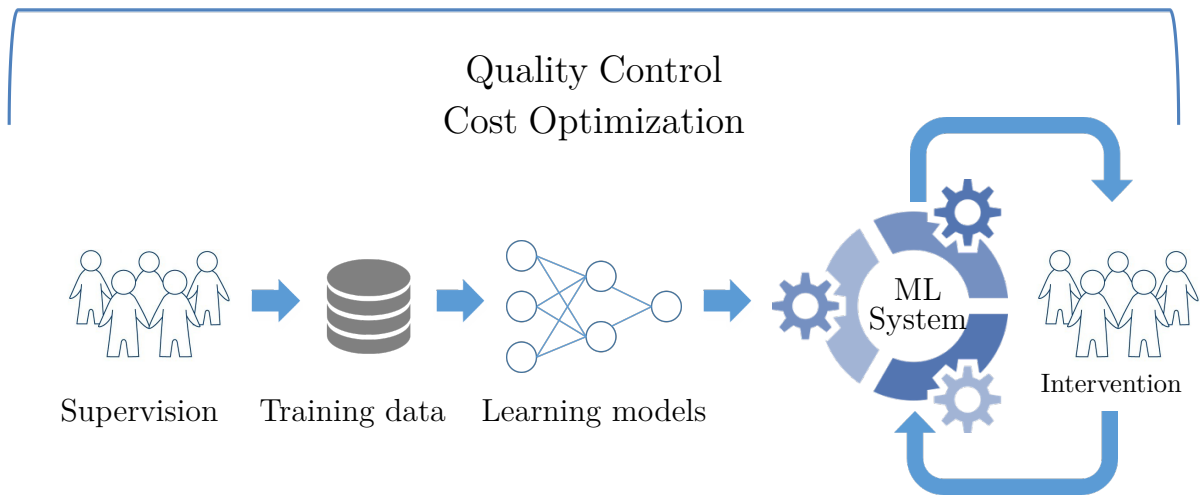


Figure 1.1: *Hybrid Human-Machine Learning Systems*

Integrating human intelligence for supporting machine learning algorithms and systems has been challenging due to the fact that large-scale and real-time data collection requires a high number of people available to participate in the process. Until a decade ago, data labels for learning algorithms were commonly provided in research labs by experts and trained individuals. The expert feedback although accurate, posed serious limitations to the size of datasets and models. With the rapid advances in crowdsourcing and human computation [65, 91] in the last years, on-demand human feedback has been facilitated by various crowdsourcing platforms like Amazon Mechanical Turk [1], Upwork [5], and CrowdFlower [3]. The main advantage of these platforms is that they provide an online human computation marketplace of a large number of workers which was not feasible in the past. Typically, human input is acquired through individual micro-tasks specifically designed for the purpose of collecting training data or providing feedback to a current system. The paradigm created practical opportunities for introducing novel hybrid systems in various research areas like machine learning (*e.g.* CrowdSynth [70], Flock [28], Gestalt [123]), databases (*e.g.* CrowdDB [50], Deco [122], Qurk [106]), and Web search and information retrieval (*e.g.* CrowdSearcher [21], Aardvark [63]).

The shift to crowdsourcing as a new paradigm for outsourcing micro-tasks to a large crowd of people instead of experts introduces new challenges in terms of (i) *quality control* [10, 92, 67] as well as (ii) *cost optimization* [75, 61, 151]. The two challenges are inherent for heterogeneous marketplaces where people have different skills and work



| | | |
|-------------------|---------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| Domain |  MODELS Supervision |  SYSTEMS Intervention |
| Problem | | |
| Quality Control | Label Aggregation | System Troubleshooting |
| Cost Optimization | Crowd Access Optimization Budgeted Learning & Feature Selection | |

Table 1.1: *Quality control and cost optimization problems*

quality. Regardless of the type of task, crowdsourcing work introduces noise either due to task subjectivity, ambiguous task design, or human error [11], which motivates the design of new algorithms and models that can either prune noisy answers (*i.e.* known as *spam detection*) or accordingly interpret answers based on the worker and task properties (*i.e.* known as *label aggregation*). In addition, crowdsourcing work comes at a given cost often measured in terms of the worker payment in paid crowdsourcing platforms [1, 5, 3] or the time that workers spend in citizen science platforms [2, 4]. For collecting large-scale supervision data or running long-term intervention processes, the crowdsourcing cost is significant and oftentimes a key feasibility factor for many projects. Therefore, the problem of cost optimization is longstanding in systems driven or enhanced by human computation.

This thesis studies the quality control and optimization challenges in crowdsourcing in the context of building hybrid human-machine learning algorithms and systems. As shown in Figure 1.1, both challenges are present throughout the whole lifecycle of such systems. Therefore, our main message is that measures for tackling these problems should be employed as umbrella activities in various stages of training and refining machine learning models as well as integrating such models into systems. Table 1.1 depicts a high-level overview of the problem space. Next, we provide an introductory definition of the distinct problems that the dissertation solves in this area and how they contribute towards improving the quality of current learning models and systems. While a few of them are well-known in the literature (*e.g.* label aggregation), others are new to the community and their characterization is also part of the thesis’ contribution.

1.1.1 Human supervision of machine learning models

Problem 1.1 (LABEL AGGREGATION). *Given a set of labels $\{x_1, \dots, x_{|W|}\}$ generated by $|W|$ crowdsourcing workers for a task Y , the goal is to find a high-quality prediction of the outcome of task Y by aggregating the worker labels.*

Label aggregation is one of the most studied problem in crowdsourcing as it provides the fundamental building block for ensuring quality while interpreting human-generated data. The problem is encountered both in the training and the testing stage of algorithms that make predictions from crowdsourced data. It applies to multiple types of prediction tasks like classification, regression, clustering etc. The thesis focuses mostly on classification tasks although the overall ideas we propose can be adapted to other types of predictions.

An important part of the problem is designing a customized model that encodes the dependencies between the various variables (*i.e.* tasks, workers etc.) and can be leveraged to infer the true value of the tasks. Traditional approaches [67, 35, 128, 176] seek to model the individual performance of workers with respect to the task which is crucial for accurately weighing decisions or excluding low-quality work. However, in real-world use cases of crowdsourcing, individual worker representations are not sufficient. Due to high data sparsity and power-log work distribution, estimating accurate worker parameters becomes challenging. Moreover, the full independence assumptions that generally accompany the individual models may result to wrong predictions or misleading confidence values associated to the predictions. In this thesis, we look at these two additional challenges, namely managing data sparsity and modeling dependencies across workers.

Problem 1.2 (CROWD ACCESS OPTIMIZATION). *Given a task Y that can be solved by aggregating labels provided by a set of crowdsourcing workers W under a crowd access budget constraint B , the goal is to find the optimal set of workers S that satisfies the budget constraint and maximizes the quality of predicting Y .*

The problem is differently known in the literature as the *task assignment* problem [75, 61, 151]. Depending on the application, the definition may take into account varying worker costs and notions of expertise / performance which naturally become part of the optimization problem. Here, we study the problem in the perspective of maximizing the quality of predictions in the testing stage based on a previously trained aggregation model of diverse workers both in terms of cost and expertise.

We observe that the challenges of crowd access optimization are tightly coupled with those of label aggregation. For example, poor estimation of workers' performance can prevent optimization algorithms to pick the best set for a task. Most importantly, potential dependencies or correlations between worker answers need to be carefully considered while accessing the crowd so that the budget is not unnecessarily spent in collecting multiple times similar or dependent insights.

Problem 1.3 (BUDGETED LEARNING AND FEATURE SELECTION). *Given a feature-based classification task Y that can be solved via a set of N candidate features $X = \{X_1, \dots, X_N\}$ with unknown labels that can be acquired through crowdsourcing, the goal is to learn a classification model θ under a budget constraint B that can make high-quality predictions of Y using only $K < N$ most informative features.*

The problems of *budgeted learning* (training models under hard budget constraints) [41, 40, 103] and *feature selection* (choosing the most informative features for prediction) [57, 58] have previously been studied orthogonal to each other in a non-crowdsourcing setting which assumes perfect feature labels (*e.g.* expert labels). Here, we focus our interest in applications where feature labels are collected at a given cost from possibly noisy crowdsourcing work.

Differently from crowd access optimization, this problem arises in the training stage and aims at early planning the feature label acquisition in order to learn good prediction models under two different but related budget constraints. The first budget constraint B is relevant to the amount of data that needs to be acquired for training such models. The second budget constraint K , is relevant to the amount of data (*i.e.* number of features) that will be used in the testing / prediction stage in order to make a single prediction from the model. The main motivation for satisfying both constraints comes from the fact that bootstrapping new datasets for training new prediction models with a large number of candidate features is expensive and oftentimes not affordable. However, planning for both budget limitations during training time is challenging as budget allocation decisions (*i.e.* which feature label to collect next) need to be taken based on insufficient data and therefore unknown / uncertain feature parameters.

1.1.2 Human intervention on machine learning systems

Problem 1.4 (SYSTEM TROUBLESHOOTING). *Given a component-based machine learning system that consists of N components $C = \{c_1, \dots, c_N\}$, the goal is to analyse system failures and understand the impact of component improvements / fixes in recovering failures and improving the overall system quality.*

Failures in machine learning applications can cause unpleasant effects on users' experience and therefore reduce their trust towards such systems. Hence, understanding failures and identifying potential fixes that can help the system improve is a problem of critical importance. Our goal is to assist system designers in deciding which components need to be fixed first in order to fix the system output to the user. In this thesis,

we start by characterizing the intrinsic challenging nature of this problem. In particular, we focus on systems composed of multiple machine learned components that work together to solve a common task. This integrated architecture is motivated by various benefits related to component reusability and maintenance as well as work division among machine learning experts. We assume that the architecture of the systems we study is *modular* and their input / output is *human-interpretable*.

To analyse the impact of component fixes in the overall system quality, system designers need to simulate or even build improved component states. This is oftentimes infeasible or expensive to implement. Therefore, we study the opportunities of involving humans in the loop for correcting the output of components. We identify the challenges of building a hybrid human-machine framework for this purpose as a valuable tool for system designers to make system-level error analysis and troubleshooting.

1.2 Background

Quality Control Quality control in human computation refers to the process of reviewing and evaluating crowdsourcing work in order to ensure that aggregated results meet given quality requirements. Research efforts in this area have characterized quality control along two main dimensions: (i) worker profiles and (ii) task design [10]. Worker profiling affects or is part of a set of activities like spam detection [159, 115, 127], expertise retrieval [16, 22] as well as incentivizing and training workers [142, 44, 64, 60, 23]. The purpose of task design quality control on the other hand is to develop appropriate task definitions that can improve workers' efficiency or their collaboration. To this end, task design encompasses various practices of human-computer interaction and collaborative work [88, 84, 82] that can help crowdsourcing requesters to adjust task instructions and visual representations (GUI design), task granularity, and worker interaction.

Our work on quality control is particularly centered around worker profiling. However, in contrast to previous work which leverages only the individual worker profiles [67, 35, 128, 176], we propose to leverage possible groupings or correlations between crowd worker answers. This coarse-grained view on the crowd helps us to avoid overconfident or wrong predictions that can be generated from models built upon insufficient or sparse data. In this context, other group-based approaches [156, 94] have also investigated similar ideas on categorizing workers according to their error confusion matrix with respect to the task.

Cost Optimization For most of the crowdsourcing use cases, cost optimization and quality control are tightly coupled to each other. In a broad perspective, optimization techniques either minimize the cost of achieving required quality standards or they maximize quality under given hard budget constraints. Such techniques can be divided in three directions: (i) task assignment [75, 61, 151] (ii) active and budgeted learning for crowdsourcing [110, 98, 166, 175], and (iii) workflow design for complex tasks [32, 149]. The work presented here mostly contributes in the first two directions.

The main theme in task assignment techniques is expertise matching for targeting tasks to the most expert workers. We observe that alongside expertise, yet another dimension to be considered is answer *diversity*. Diversity is important in budget-limited task assignment as redundant budget allocations (*i.e.* assigning the same task to multiple workers) do not reveal any new information if expert answers are correlated. This phenomena was emphasized by Surowiecki in [145] stating that the best answers are achieved from discussion and contradiction rather than agreement and consensus. Other relevant studies in management science also emphasize diversity [62, 90] and define the notion of types to refer to forecasters that have similar accuracies and high error correlation.

The problem of active learning for crowdsourcing is closely related to the traditional active learning problem formulation [138] where the goal is to reduce the cost of data collection for a learning algorithm by selectively choosing the data instances to be labeled. When data instances are labeled by humans, labels may be noisy (due to human error or subjectivity) which increases the cost of data collection as multiple answers need to be collected for the same example. The trade-off between selecting multiple labels for the same instance and selecting labels for multiple instances is thoroughly studied in [99, 98]. In this thesis, we study an extended formulation of the problem where feature labels of the data instances are unknown and are also collected via crowdsourcing.

Hybrid Human-Machine Learning Systems Learning from crowdsourced data labels is the first step of bootstrapping prediction algorithms from human-generated input. However, learning algorithms often make mistakes or are not able to correctly handle all the possible usage scenarios [135]. When several algorithms are deployed together into a single system in the form of machine learning components, the space of possible errors becomes broader. Maintaining and further advancing such systems requires continuous human intervention. This type of interaction is present or possible to integrate for numerous puposes like system

evaluation [12], troubleshooting and failure prediction [172, 119] and taking over complex tasks that automated machines cannot handle yet.

Designing hybrid systems poses several challenges which are mainly related to the communication process between people and learning systems. Therefore, a vast amount of work has been focused on interactive, human-interpretable and human-centered machine learning methods [162, 13, 80, 155, 54]. These properties are indeed important prerequisites to envision collaborative human-in-the loop frameworks that can assist current intelligent systems.

1.3 Contribution

The Access Path model We propose a new crowd model named as the Access Path model (APM) for the purpose of improving the quality of label aggregation in crowdsourcing. The APM makes use of the *access path* notion as an alternative way of retrieving an answer from the crowd. The configuration of access paths can be based on various criteria depending on the task: (i) workers demographics (*e.g.* profession, group of interest, age) (ii) the source of information or the tool that is used to find the answer (*e.g.* phone call vs. web page, Bing vs. Google) (iii) task design (*e.g.* time of completion, user interface) (iv) task decomposition (*e.g.* part of the answers, features). Therefore, the model explores crowd diversity not on the individual worker level but on the access path level which represents the common dependencies of workers while performing a task. This design can be applied even if the data is sparse and crowd workers are anonymous. We show that predictions based on this model are not only more accurate but they also map to realistic confidence values.

Furthermore, we employ the Access Path model to seamlessly tackle the problem of crowd access optimization along with label aggregation. For this purpose, we leverage highly efficient greedy algorithms with strong guarantees which use a submodular information-theoretic objective for crowd access optimization. Our optimization scheme plans the number of workers to be asked within an access path which means that crowd access is also centered around groups rather than individuals. As a result, our crowd access approach is aware of worker correlations and able to handle situations of low individual worker availability. These properties then enable the Access Path model to make robust predictions with lower cost.

B-LEAFS In the context of cost optimization for training new machine learning models, we introduce a novel algorithm B-LEAFS, to *jointly* tackle the problems of **Budgeted Learning** and **Feature Selection** for training and testing feature-based classifiers that are robust to noisy feature labels. The purpose of the algorithm is to efficiently collect feature label data for training classifiers under two types of budget constraints: (i) training phase budget constraints (the total cost of collected feature labels), and (ii) testing phase budget constraints (the number of features that can be used for prediction in the training phase). We adapt B-LEAFS for both the Naïve Bayes model and the Access Path model in order to support both noisy and non-noisy labels.

B-LEAFS operates in a Bayesian framework, and maintains posterior distributions over all model parameters, thereby enabling us to capture the uncertainty in the model parameters about individual features. The algorithm makes greedy decisions for selecting the next feature label to acquire by exploiting the submodularity of information gain from a feature, conditioned on the current state of learning. In addition, it effectively balances exploration and exploitation by employing Thompson sampling techniques [148]. Our extensive experiments on various datasets and noise regimes show that models constructed based on data collected from B-LEAFS can make better predictions with limited budget.

Human-in-the-loop troubleshooting First and foremost, we define the problem of troubleshooting component-based machine learning systems which we observe to have intrinsic characteristics not present in other analogous problems (*e.g.* troubleshooting physical devices or software systems). Next, we envision an innovative troubleshooting framework for component-based machine learning systems that employs human feedback for diagnosing and fixing system failures. This framework devises an innovative crowdsourcing methodology that combines human input with system execution in the form of crowdsourced component fixes to diagnose the output of individual components and measure the benefit of the proposed fixes on the overall system performance.

Human intervention is crucial to our idea as it can simulate improved component output that cannot be produced otherwise without significant efforts from system developers. This framework can be used from system developers to either (i) perform generic *system troubleshooting* and understand how to spend their development resources towards further improving the system, or (ii) introduce *instance troubleshooting* for correcting the system output on specific input instances with human help if immediate improvement implementation is technically not feasi-

ble. We apply our approach on a real-world running system for automatic image captioning. Our analysis in this case study shows that crowd input on system evaluation and component fixes provides valuable insights to system developers and can be effectively used to diagnose and fix system failures.

1.4 Thesis overview

Chapter 2 describes the design of the Access Path model and how we employ this model to solve multiple crowdsourced prediction tasks. We describe in detail the learning and inference steps on the model and provide guidelines on viable access path configurations. Parts of this chapter have been published in the Third AAAI Conference on Human Computation and Crowdsourcing (HCOMP) [113], and it is joint work with Adish Singla, Anja Gruenheid, Erfan Zamanian, Andreas Krause, and Donald Kossmann.

Chapter 3 solves the crowd access optimization problem in the context of access path selection. After proving the submodularity property of information gain in the Access Path model, we show how this is beneficial for adapting a greedy optimization scheme with theoretical bounds. Parts of this chapter have been published in the Third AAAI Conference on Human Computation and Crowdsourcing (HCOMP) [113], and it is joint work with Adish Singla, Anja Gruenheid, Erfan Zamanian, Andreas Krause, and Donald Kossmann.

Chapter 4 presents the B-LEAFS algorithm for learning and selecting features in the Naïve Bayes and the Access Path model under budget constraints. Experimental results on noisy and non-noisy feature label settings are then presented to evaluate the effectiveness of the algorithm in diverse regimes. Parts of this chapter have been published in the Fourth AAAI Conference on Human Computation and Crowdsourcing (HCOMP) [114], and it is joint work with Adish Singla, Andreas Krause, and Donald Kossmann.

Chapter 5 introduces the problem of troubleshooting integrative machine learning systems. Next, it proposes our human-in-the-loop framework for solving this problem with the help of human input in the form of crowdsourcing micro-tasks. The chapter describes how such a framework can be used to perform system and instance troubleshooting in the context of an image captioning system. Parts of this chapter will be published in the Thirty-First AAAI Conference, and it is

joint work with Ece Kamar, Eric Horvitz, and Donald Kossmann, as part of a collaborative project with Microsoft Research Redmond.

Chapter 6 concludes this dissertation and lays out possible future directions towards understanding and strengthening the human impact in improving the quality of current intelligent systems.

2

Label Aggregation in Crowdsourcing

2.1 Overview

Label aggregation is the fundamental building block of all algorithms and models that make predictions based on crowdsourced data. Due to human error and subjectivity, collecting a single label for a task is not sufficient. Therefore, the classical approach in most crowdsourcing platforms is to collect multiple answers / labels for a given task in order to improve the quality of predictions. In the literature, this approach is referred to as *crowdsourcing redundancy* or *answer overlap*. The collected redundant labels are then aggregated into a single final answer. The purpose of quality control in label aggregation is to ensure that the final decision is of high-quality. In this thesis, we look at two quality aspects of predictions: accuracy and confidence. While accuracy measures whether the prediction matches the correct decision, confidence maps to the level of prediction certainty. In these terms, a wrong prediction with low confidence is still of a better quality than the same prediction with a high confidence because it is more informative for decision-making.

The simplest form of aggregation is *majority vote* where the final aggregated value corresponds to the opinion of the majority part of the crowd. In this case, all workers are considered to have the same work quality which makes all labels equally important. However, in reality, crowdsourcing workers have different skills and expertise, which

| | Majority Voting | Individual Models | Access Path Model |
|-----------------------------|-----------------|-------------------|-------------------|
| Diversity awareness | ✗ | ✓ | ✓ |
| Sparsity and Anonymity | ✓ | ✗ | ✓ |
| Cost-efficient optimization | ✗ | ✗ | ✓ |
| Meaningful confidence | ✓ | ✗ | ✓ |

Table 2.1: *Comparison of the APM with current approaches.*

motivated the need to build more sophisticated aggregation methods that are based on worker profiling [35, 67, 165]. These methods learn and then employ individual worker parameters as weighing factors while aggregating labels. For example, the answer of a worker with high accuracy would count more than the answer of a less accurate worker. This type of individual profiling however faces two critical real-world challenges:

1. **Worker answer correlation** — The main assumption of individual models is that the opinions of workers are all independent. Hence, retrieving the same answer from different workers significantly boosts the confidence of predictions. Nonetheless, if independence does not hold, reinforcing predictions based on dependent opinions can result to misleading predictions both in terms of accuracy and confidence. In contrary, a prediction that is boosted by a diverse set of independent opinions is less likely to be prone of correlation errors.
2. **Data sparsity** — The next assumption in employing individual models is that workers have performed sufficient work in the past so that their parameters can be accurately estimated. This is not usually the case in open crowdsourcing marketplaces. In contrary, worker participation especially in paid platforms follows a power law distribution. Therefore, parameter estimation for most of the workers becomes challenging due to insufficient data, which then leads to unreliable predictions.

To overcome these challenges, we propose a novel label aggregation model named as the Access Path Model (APM) which is a middle-ground solution between the majority vote and individual models. Table 2.1 illustrates how the APM compares to these two ends of aggregation methods.

The core idea of the model is to represent the diversity of worker answers through the *access path* notion which is defined as an alternative way of retrieving a piece of information from the crowd. For example, while evaluating the usability of a mobile

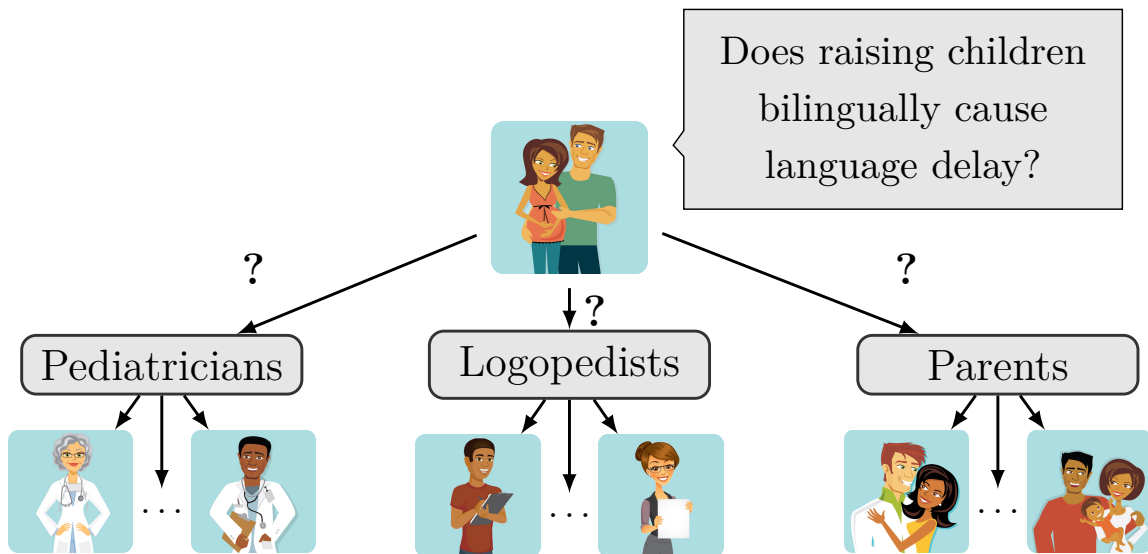


Figure 2.1: *The APM for crowdsourcing a medical question*

user interface, the designer may ask users of different age and gender in order to form an overall opinion. The assumption of the model then is that worker answers are independent across access paths but they are dependent within the same access path. In Example 2.1 we show a concrete example of applying the Access Path model to solve a real-world problem.

Example 2.1. *Peter and Aanya natively speak two different languages which they would like to teach to their young children. At the same time, they are concerned how this multilingual environment affects the learning abilities of their children. More specifically, they want to answer the question “Does raising children bilingually cause language delay?”. To resolve their problem, they can ask three different groups of people (access paths):*

| Access Path | Error rate | Cost |
|---------------|------------|------|
| Pediatricians | 10% | \$20 |
| Logopedists | 15% | \$15 |
| Other parents | 25% | \$10 |

Table 2.2: *Access path configuration for Example 1*

Figure 2.1 illustrates the given situation with respect to the Access Path Model. In this example, each of the groups approaches the problem from a different perspective

and has different associated error rates and costs. For instance, pediatricians have similar opinions due to the information that they study in medical school. The Access Path model represents these dependencies and makes sure that a prediction is better reinforced when the same answer comes from different access paths. In this specific problem, until the 90's doctors generally agreed that growing up with more than one language would lead to speech delay even though this belief was not confirmed by multilingual families [93]. In contrary, the belief was proven to be wrong afterwards and studies confirmed that bilingualism does not cause language delay. In such situations where the ground truth is not well-established, answer diversity (encoded as access path independencies in our model) is particularly important. If worker dependencies within an access path would not be taken into consideration, repeated opinions from pediatricians would lead to unrealistic confidence increase towards their opinion.

In this chapter, we present the design of the Access Path model and how it can be used to solve concrete crowdsourcing problems on label aggregation. We find that the model yields a higher quality even for sparse or anonymous data. In Chapter 3, we follow-up on this contribution and show that such a model is also beneficial for designing cost-efficient budget allocation schemes.

2.2 Related work

Individual models for label aggregation. One of the central works in the field is presented by Dawid and Skene [35]. In an experimental design where observers can make errors, the authors suggest to use the Expectation Maximization algorithm [109] to obtain maximum likelihood estimates for the observer variation when ground truth is missing or partially available. This has served as a foundation for several following contributions [67, 102, 128, 161, 165, 176], placing the algorithm of Dawid and Skene in the context of crowdsourcing and enriching it for building performance-sensitive pricing schemes. The main subject of these studies is the crowd workers, for whom repeated observations must be available.

Another interesting extension to the Dawid and Skene model that does not make full independence assumptions is presented in [81] the context of combining Bayesian classifiers (BCC models). The work proposes three extensions for modelling the correlations between individual classifiers showing correlations for “easy” and “hard” instances of data. [141] presents a dynamic version of these ideas where the confusion matrixes of workers are continuously updated. Most recently, authors in [139] prove that the individual Dawid and Skene model is equivalent to a Restricted Boltzmann Machine

with a single hidden node. Furthermore, the work also investigates on cases when the conditional independency does not hold.

For aggregation problems where no ground truth is available even for historical training data, the above EM-like techniques that are usually employed may fall into local maxima or saddle points [107]. Authors in [174] study this problem specifically in the crowdsourcing setting and they propose to perform a preceding parameter initialization step before the EM algorithm. The initialization step relies on spectral methods for estimating latent variable models [14].

Diversity for quality. The notion of diversity has been studied in various disciplines which aggregate information from possibly noisy data sources. Relevant studies in management science that emphasize diversity [62, 90] define the notion of *types* to refer to forecasters that have similar accuracies and high error correlation. These works conclude that predictions based on a large crowd of workers are more accurate if the crowd is diverse. In smaller groups, the accuracy of workers is more important than diversity.

In the context of data integration, diversity is exploited for solving problems like *truth discovery*, *copying detection* and *data freshness* [129, 43]. In particular, the authors in [124] analyse the presence of information correlation among web data sources for the purpose of data fusion.

In crowdsourcing, the diversity principle was first introduced by Surowiecki in his book “Wisdom of the crowds” [145]. The book highlights diversity and independence as critical factors in making good collective decisions claiming that crowds are smarter when there is a balance between the amount of information that is shared and held privately from its members. In the research community, recent work [156, 157] has then explored the notion of communities as groups of people that have similar error rates. This definition is a special case of an access path since correlation is a particular form of statistical dependency. The work proposes the CBCC model (Community Bayesian Combination Classifier) which builds on the previously introduced BCC models above but in addition, it leverages communities to tackle the challenge of data sparsity. In our work, we are interested in more generic forms of dependencies among workers that are not always manifested as error correlation.

Task-specific label aggregation. Except worker characteristics, another crucial aspect in aggregation are the task characteristics such as task difficulty, domain, and length. Due to technical restrictions posed from current crowdsourcing platforms, crowd requesters generally aim at posting homogeneous batches of tasks that possibly require a similar amount of effort or time so that workers can receive the same payment.

Despite these efforts, examples within the same batch may still have visible differences. For example, in a sentiment analysis task, some articles may convey a more clear sentiment than others. Others, may require more domain knowledge on the article topics to make a proper evaluation.

In [71] the authors propose a set of graphical models for detecting and handling task-specific biases that prevent common aggregation models to find out the true label when such biases are present in many of the crowdsourcing workers. Other approaches represent worker expertise mapped to the task difficulty [165, 6] as a viable feature. The model introduced in [164] uses a more extensive set of task features to model various worker error types.

Label aggregation beyond classification. Most of the label aggregation techniques that we have summarized so far, assume that crowdsourcing labels are worker answers on a common single question which usually corresponds to a classification task. Beyond classification, crowdsourced labels have been used to solve other aggregation problems which require a more holistic view on the set of collected labels. Typical examples are *clustering* or *ranking*. In these problems, information about an object may reveal further information about other objects. Such problems require different aggregation models that account for object-to-object relationships. In clustering, various works have suggested probabilistic models [53, 169] or graph-based techniques [160, 55] to reason how pairwise relationships and comparisons can be aggregated to discover underlying clusters. In ranking, the goal is to find a total order of the available objects according to users' or workers' preferences. Since these preferences are often ambiguous and hard to estimate, ranking algorithms need to solve possible disagreements that may arise not only across many workers but also within the labels of a single worker. The problem is relevant to domains like Web search and recommender systems [27] as well as query processing for sorting [105] and top-K queries [173, 56] in crowdsourced databases.

2.3 Problem definition: Label Aggregation

In this section, we define the problem of label aggregation along with its related requirements and challenges. Note that this problem is equivalent to what we presented in Chapter 1 but it is adapted in the context of probabilistic classifiers for label aggregation.

Problem 2.1 (LABEL AGGREGATION). *Given a task represented by a random variable Y , and a set of answer labels $\{x_1, \dots, x_{|W|}\}$ on this task from $|W|$ workers, represented*

2.3. Problem definition: Label Aggregation

by random variables $\{X_1, \dots, X_{|W|}\}$, the goal of the label aggregation problem is to find a high-quality prediction of the outcome of task Y by aggregating these votes.

Quality criteria. A high-quality prediction is not only accurate but should also be linked to meaningful confidence levels. Confidence is formally defined as the likelihood of the decision to be correct. This property simplifies the interpretation of predictions from a probabilistic model. For example, if a doctor wants to know whether a particular medicine can positively affect the improvement of a disease condition, providing a raw *yes/no* result answer is not sufficiently informative as it might not hold for all cases. Instead, it is much more useful to answer for example “*yes with 85% confidence*”, which is significantly different from “*yes with 55% confidence*”. Another example is betting for sport events based on crowdsourced predictions. If accurate confidence is provided, one can decide whether to place a bet or not (if the confidence is strong) or accordingly invest the right amount of money.

Requirements and challenges. To provide high quality predictions, it is essential to precisely represent the crowd. A desirable aggregation method should be able to abstract diversity through the statistical dependencies of worker answers (*i.e.* random variables) that come with the access path usage. The main aspects to be represented are:

1. The conditional dependence of worker answers within access paths given the task.
2. The conditional independence of worker answers across access paths given the task.

The answers of two workers X_i and X_j are conditionally independent given a task Y if, once the value of Y is known, the answer of X_j does not add further information about X_i and vice versa, *i.e.* $P(X_i|X_j, Y) = P(X_i|Y)$. We are interested in statistical dependencies since we want to model a broad class of relationships within a crowd. One form of these relationships is error correlation which in the most common case measures linear dependence.

Modeling such dependencies for probabilistic models is crucial for making the right predictions and, as we show in the next chapter, also for efficient optimization. These dependency aspects mimic situations when groups of people make similar decisions because they read the same media, are introduced to the same task-design *etc.* Diverse groups instead take independent and sometimes different decisions. Obviously, for certain use-cases there might be other forms of dependencies except the ones introduced by access paths. Nevertheless, the number and the types of dependencies to represent

is a trade-off between precision, generality, and complexity. Our goal is to design a model that can express the dependencies that matter so that it does not overfit to particular tasks and is also computationally efficient.

Another crucial requirement concerns the support for data *sparsity* and *anonymity*. Data sparsity is common in crowdsourcing [156] and occurs when the number of tasks that workers solve is not sufficient to estimate their errors which can negatively affect quality. In other cases, the identity of workers is not available, but it is required to make good predictions based on non-anonymized features. For example, for an election prediction task, a participant might will to share the geographical region but not the identity.

In order to fulfill these requirements, we introduce the concept of an access path as an independent way of (partially) retrieving an answer from the crowd. In Section 2.5 we describe in detail how access paths are used to define the general APM model and how this can deal with answer dependencies, sparsity, anonymity and provide high-quality at the same time.

2.4 Alternative aggregation models

Before describing the structure of the Access Path Model, we first have a look at other alternative models and their behavior with respect to quality assurance. Table 2.3 specifies the meaning of each symbol as used throughout this chapter.

2.4.1 Majority-based aggregation

Majority Vote (MV). Being the simplest of the models and also the most popular one, majority voting is able to produce fairly good results if the crowdsourcing redundancy is sufficient. Nevertheless, majority voting considers all votes as equal with respect to quality and does not have any sense of diversity. Consequently, it is not possible to integrate it with any optimization scheme other than random selection.

2.4.2 Individual models

Naïve Bayes Individual (NBI). This model assigns individual error rates to each worker and uses them to weigh the incoming votes and form a decision (Figure 2.2). In cases when the ground truth is unknown, the estimation of error rates is carried out

| Symbol | Description |
|--------------|--------------------------------------------------|
| Y | random variable of the crowdsourced task |
| X_w | random variable of worker w |
| $ W $ | number of workers |
| Z_i | latent random variable of access path i |
| X_{ij} | random variable of worker j in access path i |
| $S[i]$ | number of labels in access path i |
| N | number of access paths |
| B | budget constraint |
| D | training dataset |
| K | number of samples in the training dataset |
| $s < y, x >$ | instance of task sample in a dataset |
| θ | parameters of the Access Path Model |

Table 2.3: Symbol description for the Access Path model

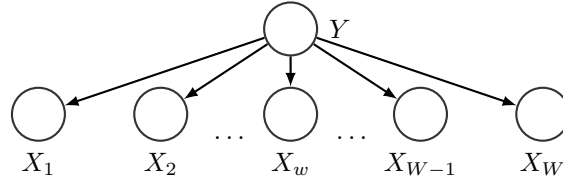


Figure 2.2: Naïve Bayes Individual - NBI.

through the EM Algorithm as in the Dawid and Skene approach [35, 109]. Aggregation (*i.e.* selecting the best prediction) is then performed through Bayesian inference. For example, for a set of votes x coming from $|W|$ different workers $X_1, \dots, X_{|W|}$ the most likely outcome among all candidate outcomes y_c is computed as:

$$\text{prediction} = \arg \max_{y_c \in Y} p(y_c|x) \text{ where } p(y_c|x) = \frac{p(y_c, x)}{\sum_{y \in Y} p(y, x)} \quad (2.1)$$

whereas the joint probability of a candidate answer y_c and the votes x_t is:

$$p(y_c, x_t) = p(y) \prod_{w=1}^{|W|} p(x_{wt}|y_c) \quad (2.2)$$

The parameters of the model correspond to the conditional probabilities $p(X_w = x|Y = y)$ estimated during the training stage for every worker. The quality of predictions highly depends on the assumption that each worker has solved a fairly sufficient number of tasks and that each task has been solved by a sufficient number of workers.

This assumption generally does not hold for open crowdsourcing markets where stable participation of workers is not guaranteed. This means that the contribution from a certain worker is either permanently missing or not available at the moment. Furthermore, even in cases of fully committed workers, this model does not provide the proper logistics to optimize the budget distribution since it does not capture the shared dependencies between workers.

Due to the Naïve Bayes inference (Equations 2.1 and 2.2) which assumes conditional independence between each pair of workers [133], predictions of this model are overconfident. Example 2.2 illustrates overconfidence in a simplified scenario.

Example 2.2. *Assume a binary task Y that has a uniform distribution $P(Y = 1) = P(Y = 0) = 0.5$. The task is solved by five workers that have symmetric true positive and true negative accuracies. Table 2.4 shows the accuracy and the respective votes of all workers on a sample task.*

| | X_1 | X_2 | X_3 | X_4 | X_5 |
|-----------------|-------|-------|-------|-------|-------|
| accuracy | 0.9 | 0.8 | 0.7 | 0.8 | 0.7 |
| label (x_i) | 0 | 0 | 0 | 1 | 1 |

Table 2.4: Accuracy rates and votes for each worker.

$$p(Y = 0, x) = 0.5 \times 0.7 \times 0.8 \times 0.9 \times 0.2 \times 0.3 = 0.01512$$

$$p(Y = 1, x) = 0.5 \times 0.3 \times 0.2 \times 0.1 \times 0.8 \times 0.7 = 0.00168$$

$$p(Y = 0|x) = 0.9 \text{ and } p(Y = 1|x) = 0.1$$

In this example, the model will output 0 as a final prediction with confidence 0.9 even though there is only one more vote in favor of this decision. If the pairwise independence across workers really holds, such a high confidence is realistic. In contrast, if for instance the answers of workers X_1, X_2, X_3 are not independent, then reinforcing the confidence through dependent observations is misleading. The direct effect of such reinforcement is that the possible answers are mapped to either very high or very low confidence which does not reflect the real confidence intervals. This discrepancy becomes more visible if the number of dependent workers within the crowd increases.

2.5 Access path based models

The crowd model presented in this section aims to fulfill the requirements specified in the definition of Problem 2.1 (LABEL AGGREGATION) and enables our method to

learn the error rates from historical data and then accordingly aggregate worker votes.

2.5.1 Access Path design

We adapted the notion of a crowd access path from the traditional access path definition in relational database systems which is one of the main pillars in query optimization for traditional databases [136]. In this context, an access path is an alternative way of retrieving information from a relation table (*i.e.* table scan, index etc.). Therefore, different access paths have a different response time but they produce identical results which means that they always return the same set of tuples. Also, physically stored data does not include any kind of uncertainty, producing therefore deterministic results.

In contrast, crowdsourced data processing deals with uncertain information coming from noisy observations. In the crowdsourcing context, access paths not only may have a different cost but they may also have different qualities which as we show in Chapter 3 makes the optimization problem more challenging. In this section, we describe how this notion can be exploited in practice in current crowdsourcing platforms along with possible use cases.

2.5.1.1 Architectural context and implications

We envision access path design as part of the quality assurance and control module for new crowdsourcing frameworks. In our case, this idea was initiated as part of the query engine in a crowdsourced database [50, 122, 106]. Here, the query executor then needs to be extended with aggregation functionalities (*probabilistic inference on the APM*) so that it can interpret the data after collecting it from the crowd. The query optimizer determines the optimal combination of access paths as we discuss in the next chapter.

The access path design can be provided by the task designer as part of the task configuration. Similarly as in a traditional database, the designer decides to add an access path as an alternative index to the crowdsourced data. This design can either correspond to groups of workers (*e.g.* based on location) or simply to groups of answers (*e.g.* based on their source of information). In fact, the access path notion is a broad concept that can accommodate various situations and may take different shapes depending on the task. We discuss possible configurations and use cases in Section 2.5.1.2.

When the access path configuration is not as intuitive, an alternative approach is to attempt discovering groups of workers from historical data. We do not cover such

techniques in this thesis but we discuss possible solutions and preliminary ideas in Section 2.5.1.3.

2.5.1.2 Access path configuration and use cases

Below we describe a non-exhaustive list of possible configurations that are easy to directly apply in current crowdsourcing platforms.

- **Demographic groups.** Common demographic characteristics (location, gender, age) can establish strong statistical dependencies of workers' accuracy [76]. Such groups are particularly diverse for problems like sentiment analysis or product evaluation and can be retrieved from crowdsourcing platforms as part of the task, worker information, or qualification tests.
- **Information sources.** For problems like data collection and integration, the data source being used to deduplicate or match records (addresses, business names *etc.*) is the primary cause of error or accuracy [124]. In our experiments for instance, we forwarded the information source to the workers by including a respective link in the task and performed additional sanity checks.
- **Task design.** In other cases, the answer of a worker may be affected by the user interface design. For instance, in crowdsourced sorting, a worker may rate the same product differently depending on the scaling system (stars, 1-10 *etc.*) or other products that are part of the same batch [118].
- **Task decomposition.** Often, complicated problems are decomposed into smaller ones. Each subtask type can serve as an access path. For instance, in the bird classification task that we study later in our experiments, workers can resolve separate features of the bird (*i.e.* color, beak shape *etc.*) rather than its category. The subtasks will then appear as separate hits and can be solved independently.

These configurations can be included in a crowdsourcing platform in the form of worker qualifications [116] or as part of the task user interface design.

2.5.1.3 Access path discovery

Access Path discovery is related to the problem of automatically detecting groups of workers with strong answer dependency from historical data. Possible helpful tools

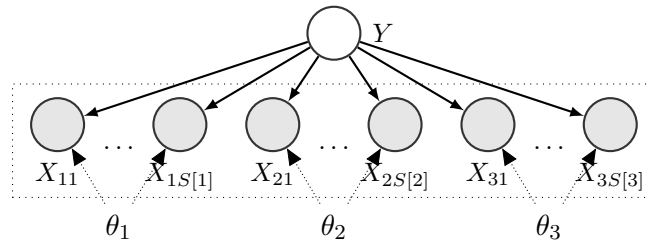


Figure 2.3: *Naïve Bayes Model for Access Paths - NBAP.*

in this regard include graphical model structure learning based on conditional independence tests [36] and information-theoretic group detection [95]. In particular, the clustering algorithm presented in [158] discovers hierarchical structures in random variables for the purpose of explaining correlations between variables.

The main challenge that these methods face in the specific domain of crowdsourcing is the sparse nature of the worker-task observations. For example, in a scenario when tasks are assigned to r different workers from a pool of $|W|$ workers in total, data sparsity is $\frac{1-r}{|W|}$. Due to the power-law work distribution among workers, this ratio is then often significantly low (even less than 2%). One related effect to this problem is that the worker similarity is hard to estimate when they have either a few or no solved tasks in common. In such circumstances, every access path discovery algorithm needs to operate under incomplete data or in addition complete the missing values [77, 78]. For heterogeneous batches of tasks, another idea would be to jointly leverage groups of workers together with groups of tasks so that the worker similarity can then be defined not in terms of tasks that workers have in common, but in terms of tasks of the same group that workers have in common.

We explored these ideas in [125] in collaboration with Matteo Pozzetti in the context of his Master Thesis project. The project highlighted the following conclusions. Current clustering algorithms (*e.g.* k-means) or hierarchical correlation explanation techniques (*e.g.* CorEx [158]) can recover possible groups in the crowd only if the data sparsity is less than 80%. For higher sparsity levels, the access path discovery error is significantly higher. In these cases, the Access Path discovery can be extended to use the task grouping and similarity to reveal worker groupings. The idea is also useful for label aggregation and budget allocation as it can differentiate workers' skills according to the task group. However, it is important to note that these extensions are beneficial only in use cases with heterogeneous tasks types. For use cases with homogeneous tasks, it is more efficient to cluster the workers only from their answers.

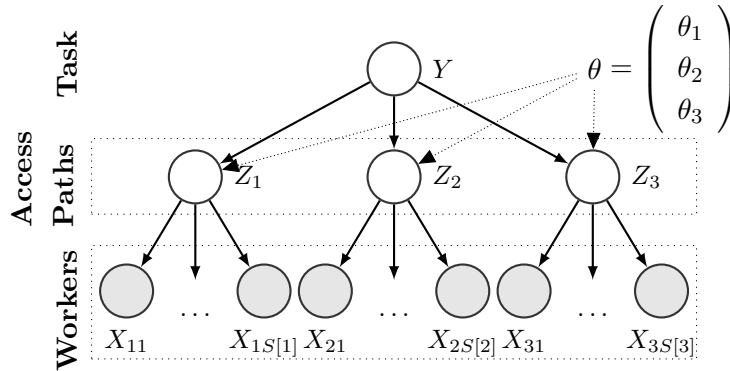


Figure 2.4: Bayesian Network Model for Access Paths - APM.

2.5.2 Access Path based models

Access Path based models group the answers of the crowd according to the access path they originate from. We first describe a simple Naïve Bayes version of such a model and then elaborate on the final design of APM.

2.5.2.1 Naïve Bayes for Access Paths (NBAP)

For correcting the effects of non-stable participation of individual workers we first consider another alternative, similar to our original model, presented in Figure 2.3. The votes of the workers here are grouped according to the access path. For inference purposes then, each vote x_{ij} is weighed with the average error rate θ_i of the access path it comes from. In other words, it is assumed that all workers within the same access path share the same error rate. As a result, all votes belonging to the same access path behave as a single random variable, which enables the model to support highly sparse data. Yet, due to the similarity with NBI and all Naïve Bayes classifiers, NBAP cannot make predictions with meaningful confidence especially when there exists a large number of access paths.

2.5.2.2 Access Path Model overview

Based on the analysis of previous models, we propose the Access Path Model as presented in Figure 2.4, which shows an instantiation for three access paths. We design the triple $\langle \text{task, access path, worker} \rangle$ as a hierarchical Bayesian Network in three layers.

| | | | |
|-------|-------------------------|-----------------------------|------------|
| | | Y | |
| | | 1 | 0 |
| | | $P(Y = 1)$ | $P(Y = 0)$ |
| | | Z_i | |
| Y | 1 | | 0 |
| 0 | $P(Z_i = 1 Y = 0)$ | $1 - P(Z_i = 1 Y = 0)$ | |
| 1 | $P(Z_i = 1 Y = 1)$ | $1 - P(Z_i = 1 Y = 1)$ | |
| | | X_{ij} | |
| Z_i | 1 | | 0 |
| 0 | $P(X_{ij} = 1 Z_i = 0)$ | $1 - P(X_{ij} = 1 Z_i = 0)$ | |
| 1 | $P(X_{ij} = 1 Z_i = 1)$ | $1 - P(X_{ij} = 1 Z_i = 1)$ | |
| | | | |

Figure 2.5: Parameters θ of the Access Path Model.

Layer 1. Variable Y in the root of the model represents the random variable modeling the real outcome of the task.

Layer 2. This layer contains the random variables modeling the access paths Z_1, Z_2, Z_3 . Each access path is represented as a latent variable, since its values are not observable. Due to the diverging tree pattern, every pair of access paths is conditionally independent given Y while the workers that belong to the same access path are not. The conditional independence is the key of representing diversity by implementing therefore various probabilistic channels. Their task is to distinguish the information that can be obtained from the workers from the one that comes from the access path.

Such enhanced expressiveness of this auxiliary layer over the previously described NBAP model avoids overconfident predictions in the following way. Whenever a new prediction is made, the amount of confidence that identical answers from different workers in the same access path can bring is first blocked by the access path usage (*i.e.* the latent variable). If the number of agreeing workers within the same access path increases, confidence increases as well but not at the same rate as it happens with NBI. The main reason is that additional workers contribute only with their own signal, while the access path signal has already been taken into consideration. The direct effect of this design is that the confidence of possible answers is more realistic and does not map to extreme values only. In terms of optimization, this property of the APM model makes a good motivation for combining access paths within the same plan and not limiting the accesses to the same channel. For our experiments, we focus

on binary Z variables but the model can be easily adapted to higher cardinalities.

Layer 3. The lowest layer contains the random variables X modeling the votes of the workers grouped by the access path they are following. For example, X_{ij} is the j -th worker on the i -th access path. The incoming edges represent the error rates of workers conditioned by their access paths.

The choice of using a Bayesian Network to design the Access Path Model is based on the ability of such networks to explicitly manage and quantify uncertainty. This is a general-purpose design and can be customized to the crowdsourced task by accordingly choosing the type and cardinality of the variables. The acyclic structure of the model supports efficient optimization schemes as we will show in Chapter 3. Nevertheless, note that the assumption of conditional independence of worker answers across access paths is also a simplification design decision. Full independence of opinion is often hard to achieve and one can think of introducing more fine-grained dependencies in the model which should be handled with caution so that (i) the problem does not become intractable, and (ii) the model does not overfit for the training data.

2.5.2.3 Parameter learning

The purpose of the training stage is to learn the parameters of the model, *i.e.* the conditional probability of each variable with respect to its parents that are graphically represented by the network edges. We will refer to the set of all model parameters as θ . Figure 2.5 shows an example of θ for a pure binary setting of the network. Given a training dataset D with historical data of the same type of task, the goal of the parameter learning stage is to find the maximum likelihood estimate θ_{MLE} for the training set. In practice, this means finding the parameters that are the best probabilistical description of the data. By definition, θ_{MLE} is a Maximum Likelihood Estimate for θ if $\theta_{MLE} = \arg \max_{\theta} p(D|\theta)$. For a training set containing K samples:

$$p(D|\theta) = \prod_{k=1}^K p(s_k|\theta) \quad (2.3)$$

If all the variables $\langle Y, Z, X \rangle$ were observable then the likelihood of a sample s_k training set given θ would be:

$$p(s_k|\theta) = p(y_k|\theta) \prod_{i=1}^N \left(p(z_{ik}|y_k, \theta) \prod_{j=1}^{S_k[i]} p(x_{ijk}|z_{ik}, \theta) \right) \quad (2.4)$$

where $S_k[i]$ is the number of votes in access path Z_i for the sample. Since maximizing the likelihood estimation is equivalent to minimizing the negative log likelihood, θ_{MLE}

can be rewritten as:

$$\theta_{MLE} = \arg \min_{\theta} - \sum_{k=1}^K \log p(s_k | \theta) \quad (2.5)$$

For this setting, the estimate for $\theta_{Z_i|Y}$ can be computed by taking the derivative on both sides in order to find the inflection point:

$$\frac{\partial \log p(D|\theta)}{\partial \theta_{Z_i|Y}} = \sum_{k=1}^K \frac{\partial \log p(z_{ik}|y_k)}{\partial \theta_{Z_i|Y}} \quad (2.6)$$

For fully observable Z_i the best estimate would be:

$$\theta_{Z_i=z|Y=y} = \frac{\sum_{k=1}^K \delta(z_{ik} = z, y_k = y)}{\sum_{k=1}^K \delta(y_k = y)} \quad (2.7)$$

Here, $\delta()$ is an indicator function which returns 1 if the training example fulfills the conditions of the function, and 0 otherwise. Since in our model Z_i is not observable, counting with the indicator function is not possible. For this purpose, we apply the Expectation Maximization algorithm [109]. Next, we show the instantiation of one iteration of the EM algorithm for our model.

E-step: Calculates the expected value for the log likelihood of the latent variables given the current θ' . For a binary variable Z_i , for each sample this would be:

$$E[Z_{ik} = z] = \frac{p(z_{ik} = z, y_k, x_k | \theta')}{\sum_{z' \in \{0,1\}} p(z_{ik} = z', y_k, x_k | \theta')} \quad (2.8)$$

M-step: Recomputes θ by maximizing the expected log likelihood found on the E-step. Differently from what is shown in Equation 2.7, the counter for the latent variable is replaced by its expected value.

$$\theta_{Z_i=z|Y=y} \leftarrow \frac{\sum_{k=1}^K \delta(y_k = y) E[Z_{ik} = z]}{\sum_{k=1}^K \delta(y_k = y)} \quad (2.9)$$

$$\theta_{X_{ij}=x|Z_i=z} \leftarrow \frac{\sum_{k=1}^K \delta(x_{ijk} = x) E[Z_{ik} = z]}{\sum_{k=1}^K E[Z_{ik} = z]} \quad (2.10)$$

Notice that Equation 2.10 models the situation when the votes are always ordered by the worker identifier. This scheme works if the set of workers involved in the task is sufficiently stable to provide enough samples for computing the error rates of each worker (i.e. $\theta_{X_{ij}|Z_i}$) and if the worker identifier is not hidden. Since in many of the crowdsourcing applications (as well as in our experiments and datasets) this is not always the case, we share the parameters (error rates) of all workers within an access path:

$$\theta_{X_{ij}=x|Z_i=z} \leftarrow \frac{\sum_{k=1}^K \frac{\sum_{j=1}^{P[i]} \delta(x_{ijk} = x)}{P[i]} E[Z_{ik} = z]}{\sum_{k=1}^K E[Z_{ik} = z]} \quad (2.11)$$

This enables us to later apply on the model an optimization scheme agnostic about the identity of workers. The generalization is optional for APM and obligatory for NBAP.

The two EM steps are repeated until the algorithm converges and the parameters in θ do not change anymore. If there are at most M answers per access path and the algorithm converges after I steps, the complexity of parameter learning for APM is $\mathcal{O}(KNMI)$.

Training cost analysis. The main prerequisite for applying the Access Path Model as a supervised learning technique is that the task should be repetitive to enable parameter learning. In crowdsourced databases, this requirement is relevant to statistics management for query optimization. The amount of data needed to train APM is significantly lower than what individual models require which results in a faster learning process. The reason is that APM can benefit even from infrequent participation of individuals X_{ij} to estimate accurate error rates for access paths Z_i . Moreover, sharing the parameters of workers in the same access path reduces the number of parameters to learn from W for individual models to $2N$ for APM which is at least three orders of magnitude lower. In all our experiments and use-cases we observe that the benefit from this generalization is higher than the respective loss. However, it is possible to adjust the model accordingly if it is known that certain workers have a stable participation. In this case, it is recommended to keep such workers distinct and learn their individual errors. The optimization scheme should also be adjusted to explicitly decide whether to include these cases in the final plan.

2.5.2.4 Inference

After parameter learning, the model is used to infer the answer of a task given the available votes on each access path. As in previous models, the inference step computes the likelihood of each candidate outcome $y_c \in Y$ given the votes in the test sample x_t .

$$\text{prediction} = \arg \max_{y_c \in Y} p(y_c | x_t) \text{ where } p(y_c | x_t) = \frac{p(y_c, x_t)}{\sum_{y \in Y} p(y, x_t)} \quad (2.12)$$

Since the test samples contain only the values for the variables X , the joint probability between the candidate outcome and the test sample is computed by marginalizing over all possible values of Z_i as in Equation 2.13. For a fixed cardinality of Z_i , the overall complexity of inferring the most likely prediction is then $\mathcal{O}(NM)$.

$$p(y, x_t) = p(y) \prod_{i=1}^N \left(\sum_{z \in \{0,1\}} p(z|y) \prod_{j=1}^{S_t[i]} p(x_{ijt}|z) \right) \quad (2.13)$$

Besides inferring the most likely outcome, we are also interested in the confidence of the prediction. In other words, we would also like to know what is the likelihood that the prediction is accurate. For all models except Majority Vote, confidence corresponds to $p(\text{prediction}|x_t)$ computed as in Equation 2.12. Marginalization in Equation 2.13 is the technical step that avoids overconfidence by smoothly blocking the confidence increase when similar answers from the same access path are observed.

2.6 Experimental evaluation

We evaluated our work on four real-world data-sets, covering many of the use cases described in Section 2.5.1. The main goal of the experiments is to validate the proposed model. We compare our approach with other state of the art alternatives and results show that label aggregation with the Access Path model can make high-quality predictions even when the data is sparse.

2.6.1 Metrics

The comparison is based on two main metrics: *accuracy* and *negative log-likelihood*. Accuracy corresponds to the percentage of correct predictions. Negative log-likelihood is computed as the sum over all test samples of the negative log-likelihood that the prediction is accurate. Hence, it measures not only the correctness of a model but also its ability to output meaningful confidence.

$$-\text{logLikelihood} = - \sum_{s_t} \log p(\text{prediction} = y_t | x_t) \quad (2.14)$$

The closer a prediction is to the real outcome the lower is its negative log-likelihood. For example, imagine a model that makes two correct binary predictions: a prediction of high-confidence 90%, and another one of low-confidence 60%. The high-confidence decision has a negative log-likelihood of $-\log(0.9) \approx 0.05$, while for the low-confidence one is $-\log(0.6) \approx 0.22$. Thus, a desirable model should offer *low* values of negative log-likelihood which intuitively map to high confidence for accurate predictions and low confidence for those being inaccurate. Due to the logarithmic nature of the metric, the penalty of the erroneous predictions is higher than the reward of accurate ones.

2.6.2 Dataset description

All the following datasets come from real crowdsourcing tasks and consist of votes gathered from people. For experiments with restricted budget, we repeat the learning and prediction process several times via random vote selection and k-fold cross-validation.

CUB-200. The dataset [163] is built in the context of a large-scale data collection for feature-based classification of bird images on Amazon Mechanical Turk (AMT). Since this is a difficult task even for experts, the crowd workers are not directly asked to determine the bird category but whether a certain attribute is present in the image or not. Each feature (*e.g.*, yellow beak) brings a piece of information for the problem and we treat them as access paths. The dataset contains 5-10 answers for each of the 288 available feature. We keep the cost of all access paths equal as there was no clear evidence of features that are more difficult to distinguish than others. The total number of answers is approximately 7.5×10^6 for 50 candidate photos from each of the 50 bird species.

As in these experiments workers solve only a part of the task, it is not possible to apply Majority Vote or the NBI model. For a basic comparison we performed additional experiments on AMT based on the same images as the original dataset. Each hit consisted of five photos to be classified as well as a good quality sample photo from the real species (the latter was included to train the workers). Ten workers were asked to decide whether the bird in the photo belongs to the same species as the one in the sample.

MedicalQA. We gathered 100 medical questions and forwarded them to AMT. The turkers were asked to answer the questions after reading in specific health forums. The forums considered for this study belong to one of the categories in Table 2.5 which we then design as access paths. 255 workers participated in our experiment. Workers could solve all questions, but each question could be solved by a certain worker through one access path only. The origin of the answer was checked via an explanation url provided along with the answer as a sanity check. The tasks were paid equally to prevent the price of the task to affect the quality of the answers. Nevertheless, for experimental purposes, we assign an integer cost of (3, 2, 1) based on the reasoning that in real life doctors are more expensive to ask, followed then by patients and common people. We collected 10 answers per access path, which resulted in 30 answers per question.

ProbabilitySports. This data is based on a crowdsourced betting competition on NFL games[126]. The participants in the competition voted on the question: “*Is the home team going to win?*” for 250 events within the season. Not all participants voted

| Description | Forums |
|---------------------------|-------------------------------------------|
| (1) Answers from doctors | www.webmd.com www.medhelp.org |
| (2) Answers from patients | www.patient.co.uk www.ehealthforum.com |
| (3) General Q&A forum | www.quora.com www.wiki.answers.com |

Table 2.5: *Access Path Design for MedicalQA dataset.*

| Dataset | Classes | Features (access paths) | Examples |
|----------|---------|-------------------------|----------|
| car | 4 | 6 | 1728 |
| derm | 6 | 34 | 366 |
| mushroom | 2 | 22 | 8124 |
| nursery | 5 | 8 | 12960 |

Table 2.6: *UCI datasets description.*

on all events and different seasons have different popularity. In total, there are 5,930 players in the entire dataset contributing with 1,413,534 bets. We designed the access paths based on the accuracy of each player in the training set which does not reveal information about the testing set samples. Since the players' accuracy in the dataset follows a normal distribution, we divide this distribution into three intervals where each interval corresponds to one access path (worse than average, average, better than average). In this configuration, access paths have a decreasing error rate. Thus, we assign them an increasing integer cost (2, 3, 4), although the competition itself was not incentivized by monetary rewards.

UCI. These datasets [96] are part of the UCI Machine Learning Repository. They are not typical crowdsourcing datasets in that they contain a single label per feature provided by experts which is considered to be correct. In this chapter, we use these datasets as a basis structure and then inject noise on the given data in order to study the impact of various noise regimes in the prediction quality. Table 2.6 shows the datasets that we used for the study. Note that these datasets are generally used as benchmarks for evaluating feature-based classifiers. Similarly to the CUB-200 dataset, we use their features as access paths.

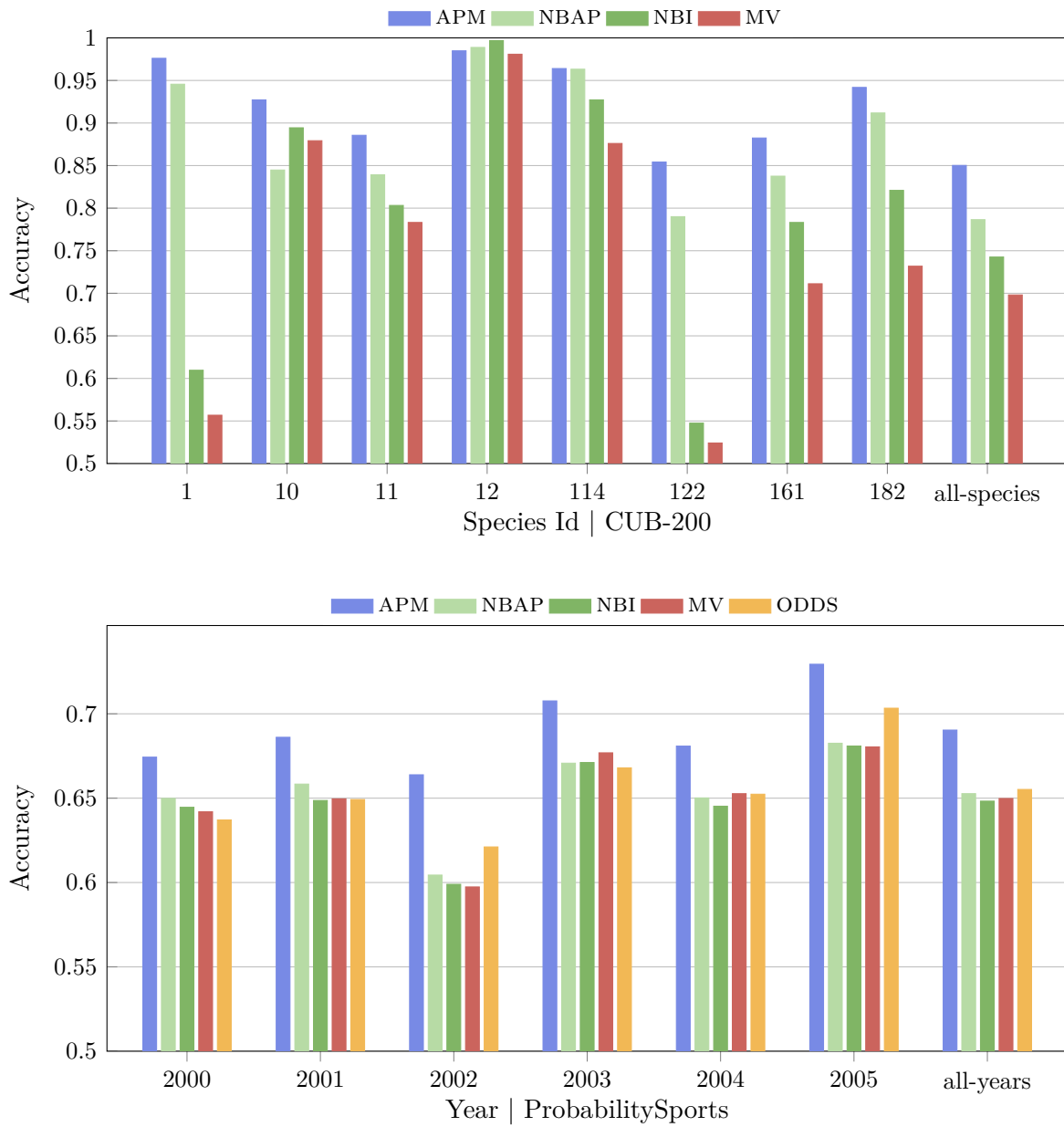


Figure 2.6: Accuracy with unconstrained budget.

2.6.3 Model evaluation on real-world predictions

For evaluating the Access Path Model independently of the optimization, we show experiments first using all available votes in the datasets and then equally distributing the budget across all access paths. The question we want to answer in this set of

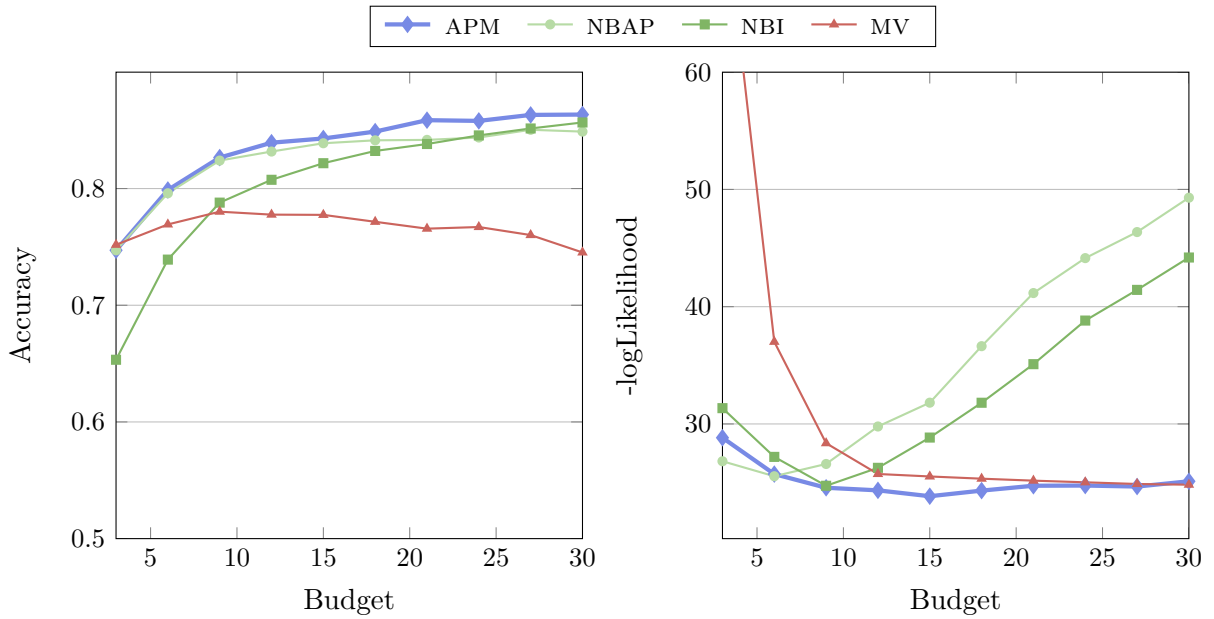


Figure 2.7: Accuracy and negative log-likelihood for equally distributed budget across all access paths in the MedicalQA dataset.

experiments is: “How robust are the APM predictions in terms of accuracy and negative log-likelihood w.r.t. state-of-the-art methods?”

2.6.3.1 Predictions with unconstrained budget

Figure 2.6 shows the accuracy of all models if the full set of available votes is used. The aim is to test the aggregation robustness of APM in case of high redundancy. As MedicalQA has relatively low redundancy compared to the other datasets and consists of only one set of questions, we will discuss it in the next experiment.

CUB-200. From this dataset we studied 50 species that belong to the same kind (e.g. different types of sparrows, seagulls etc.) and tested the efficiency of our model to distinguish between species although they might look quite similar. Note that this classification task is harder than distinguishing across different kinds of birds. For example, a seagull is a lot different from a bluebird as their features are significantly different from each other. After this comparison, it can be observed that access path models generally perform better than individual and majority models. Many species were difficult to resolve without the presence of access paths. In special cases, when the bird has a strong feature that makes it distinguishable (e.g speciesId 12, yellow headed

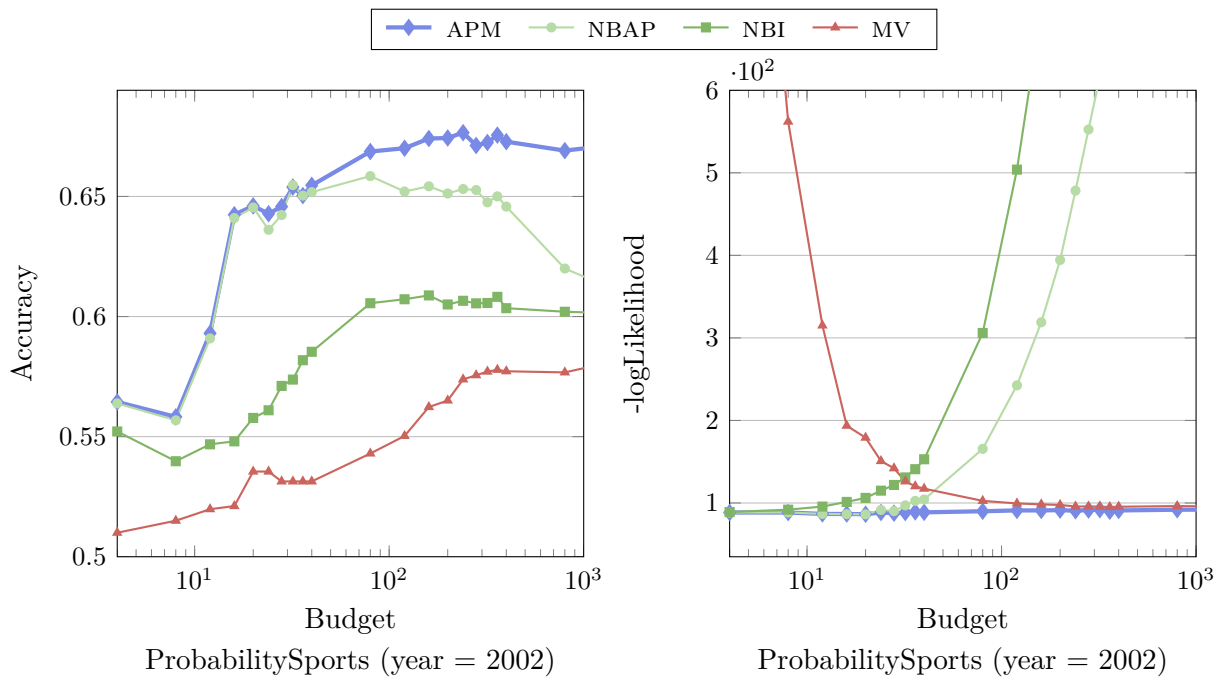


Figure 2.8: Accuracy and negative log-likelihood for equally distributed budget across all access paths in the ProbabilitySports dataset.

blackbird) there is no difference between the models. Also, often accuracy of NBI is close to MV because of the non-stable participation of AMT workers throughout all the photos from the same species.

ProbabilitySports. As expected, for the betting scenario it is challenging to improve over Majority Vote when redundancy is high (more than 1000 bets per event). Nevertheless, although our model was not specifically customized to predict betting events, we notice a 4%-8% enhancement of APM over majority while the Naïve Bayes baselines cannot achieve significant improvement. Our results cover the most popular seasons of the competition. We also show the accuracy of the odds provided by betting parties.

2.6.3.2 Predictions with constrained budget

Figures 2.7 and 2.8 illustrate the effect of data sparsity on quality. We varied the budget and uniformly distributed it across all access paths (*e.g.* for a budget $B = 30$ we randomly selected 10 votes per access path). We do not show results from CUB-200 dataset as the maximum number of votes per access path in this dataset is 5-10.

MedicalQA. The participation and the performance of Mechanical Turk workers in this experiment was generally higher than in the other datasets, which allows for a better error estimation of workers. According to workers’ feedback, the questions were interesting and they found the provided health forums informative. Hence, as shown in Figure 2.7, only for maximum redundancy ($B = 30$) NBI reaches comparable accuracy with APM although the negative log-likelihood dramatically increases. For lower budgets (*i.e.* high sparsity) NBI cannot manage to provide accurate results.

ProbabilitySports. Figure 2.8 shows that while the improvement of APM accuracy over NBI and MV is stable, NBAP starts facing the overconfidence problem while budget increases. NBI exhibits low accuracy due to very high sparsity even for sufficient budget. A further observation is the improvement of negative log-likelihood for MV which reflects the robustness of majority to provide meaningful confidence levels for high redundancy. Yet, majority fails to produce accurate predictions as it is agnostic to error rates.

2.6.4 Label noise impact on predictions

In all experiments presented so far, the amount of noise is inherent to the data as the labels were originally generated by crowdsourcing workers. In order to study the impact of various noise settings, we synthetically injected noise in the feature labels of UCI datasets. For this purpose, we experimented with two different noise regimes: *uniform* (Figure 2.9) and *biased* (Figure 2.10) noise. For these regimes we synthetically generate 5 feature labels for both the training and the testing splits of the dataset. In the uniform noise case, the label either replicates the true feature value or is uniformly picked with a varying probability (horizontal axis in all graphs) from the possible feature values different from the true value. This simulates random but non-biased worker mistakes. The biased noise regime instead imitates situations when workers could have a strong bias in consistently confusing the true value for another erroneous feature value. Note that, both noise regimes are present in real-world crowdsourcing platforms. For the Naïve Bayes baseline, we aggregate the synthetically generated labels via majority vote, and then apply a Naïve Bayes model on the aggregated data. The NO NOISE predictions are the best predictions that can be achieved when there is no crowdsourcing noise. We observe that both models can handle biased noise better than uniform noise, which is reflected in the lower prediction error for this regime. Depending on the feature cardinality, both models also can partially recover from high noise levels by probabilistically interpreting labeling errors. For example, in the uniform noise regime, if feature labels are wrong with high probability, the

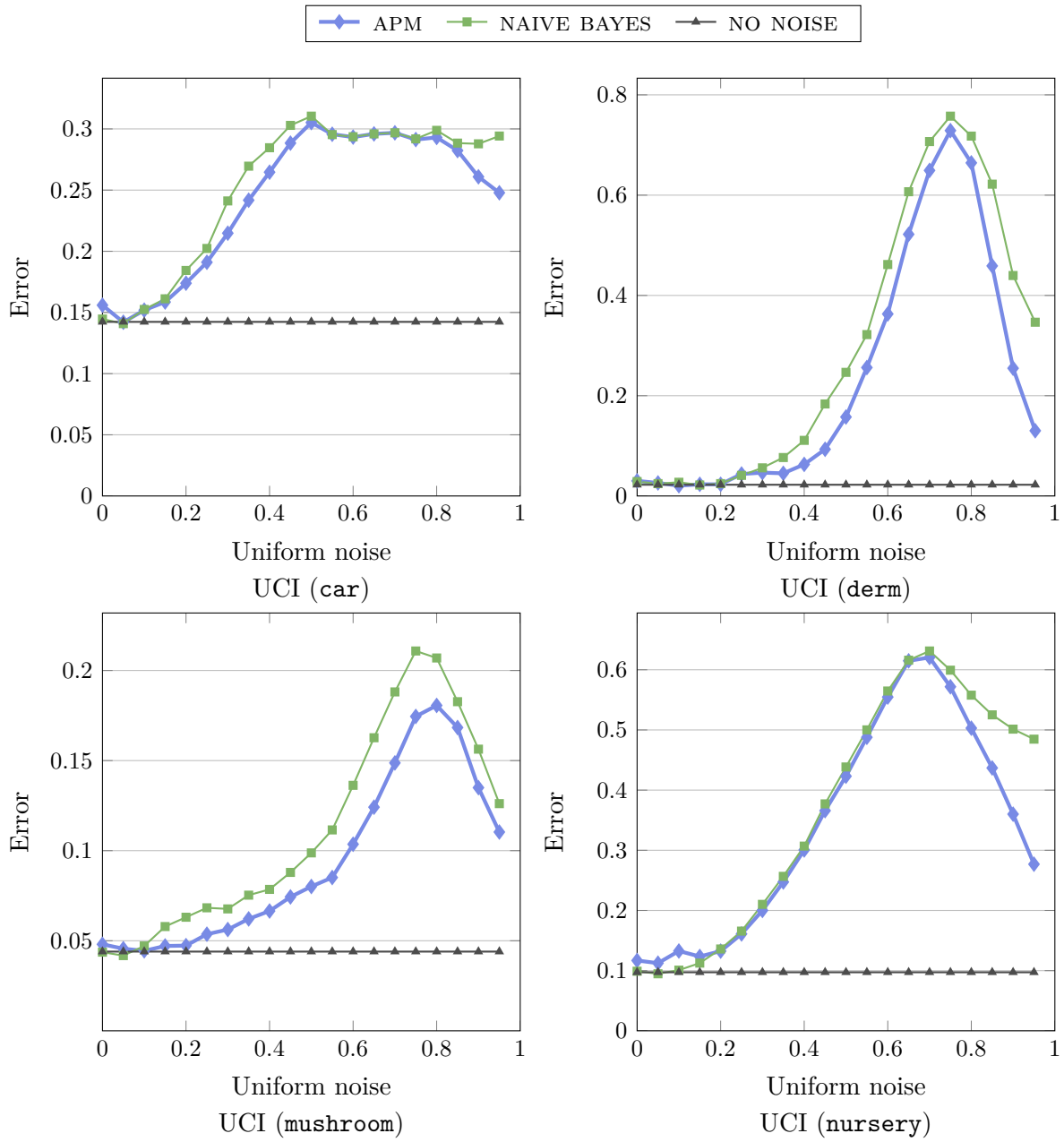


Figure 2.9: Uniform noise impact on prediction error for UCI datasets.

models are able to understand that the correct feature value is the one that is never mentioned by the workers. APM is better at interpreting these cases and it recovers faster from worker mistakes as it is able to decouple the the crowdsourcing noise from the underlying feature noise.

2.6. Experimental evaluation

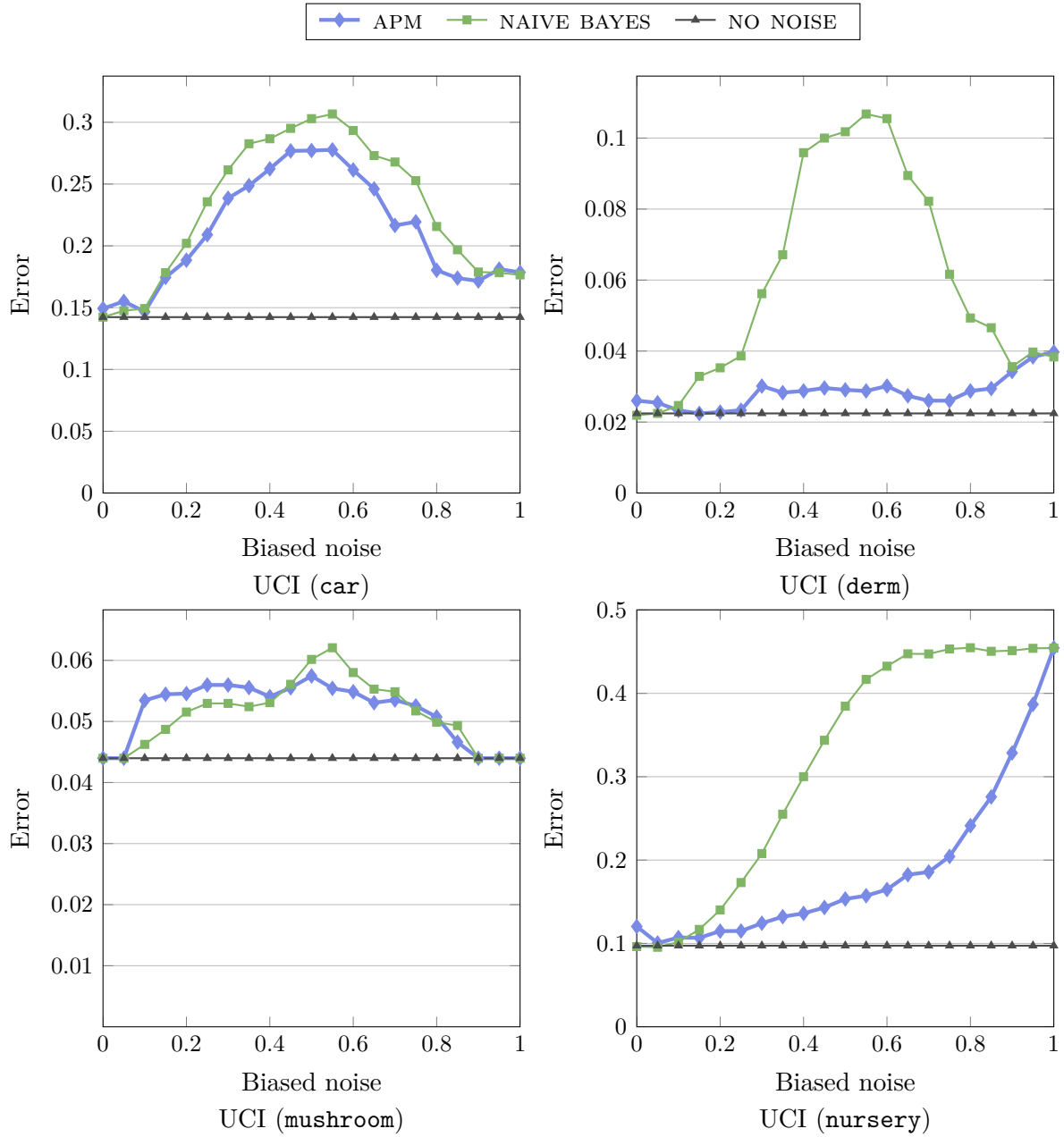


Figure 2.10: *Biased noise impact on prediction error for UCI datasets.*

2.7 Summary

In this chapter, we presented the Access Path model as a label aggregation technique that leverages correlations between worker answers to make predictions that are not only more accurate but also more meaningful in terms of confidence. Meaningful confidence is particularly beneficial when crowdsourced predictions are used for decision-making. The Access Path model distinguishes the noise that comes from the access paths from the noise induced by human error. This design property allows the model to make accurate interpretations of crowd worker labels even in the presence of noise. In addition, the model is robust to high data sparsity and worker anonymity, which is an expected and standard phenomena in crowdsourcing.

The access path notion together with the proposed model are generic tools that can be adapted to a large number of crowdsourcing problems that involve label aggregation. Current crowdsourcing platforms can make use of the model by allowing requesters to define their own access path design or by suggesting potential access paths based on historical data. In our work, we explored the idea of access paths applicable to individual tasks. Given ample data from such platforms, it is interesting to also understand how the access path design can be transferred and reused between different problems executed within the same platform and by the same crowd.

3

Crowd Access Optimization

3.1 Overview

Due to the large scale of datasets required to train machine learning models, collecting label training data is expensive both in terms of worker effort and monetary cost. For example, the creation of the ImageNet dataset [38] as an ontology of images for knowledge bases, required the crowdsourced mapping of 3.2 million images to 5247 synsets. Moreover, due to subjectivity or human error, the labeling process is usually repetitive, which means that the same label is acquired multiple times from different workers. Therefore, crowd access optimization techniques are crucial for obtaining the best quality for the cost.

In the general context, all optimization techniques aim at building appropriate budget allocation schemes which for a given budget select the set of crowdsourcing workers that are expected to provide the best answers for the task. Therefore, budget allocation is tightly dependent on previous knowledge about the expertise of crowdworkers as a prime criterion for worker selection. Previous work on budget allocation [75, 61, 151] relies on individually learning the expertise of each worker from previous data and then leveraging this information for future optimization.

In our work, we emphasize that crowd access optimization techniques face similar challenges as label aggregation, namely *worker answer correlation* and *data sparsity* as presented in Chapter 2. In label aggregation, these challenges may prevent aggregation models to provide high-quality predictions. In crowd access optimization, the same

challenges may lead to expensive data collection. We illustrate these challenges in the context of Example 2.1. In this example, parents may have a budget constraint B to find an answer to their question and need to know how many different people to ask from each access path. Every access path brings a different perspective to the problem. If parents would spend the whole budget asking people from the same access path, this strategy would not provide new insights as the acquired answers are correlated. Instead, we show that optimal strategies distribute the budget across a diverse set of access paths. Moreover, as we explain in Chapter 2, data sparsity affects the accuracy of worker expertise profiling. As most budget allocation techniques rely on these profiles, the budget may be assigned to workers which appear to be experts based on insufficient data.

Since the Access Path model can efficiently handle both these challenges, we propose an optimization scheme based on the same model, which aims at finding the best budget distribution across access paths. This implies that we shift our focus from individual *worker selection* to *access path selection*. A crowd access strategy is defined by the notion of an *access plan*, which defines how many different workers to ask within the same access path. In order to find optimal access plans for a given budget constraint, the proposed scheme uses an information-theoretic objective, capturing the reduction in uncertainty. We prove that our objective is submodular, allowing us to leverage highly efficient greedy algorithms with strong guarantees.

3.2 Related work

Crowd access optimization. The problem of finding the best crowd access plan is similar to expert selection in decision-making. However, differently from expert selection, for crowd access optimization, the assumption that the selected individuals will answer no longer holds, even in paid forms of crowdsourcing. Previous studies based on this assumption are [59, 73]. The proposed techniques are effective for task recommendation, spam detection, and performance evaluation, but they can easily run into situations of low participation and high sparsity.

Due to the current limitations of crowdsourcing platforms with respect to worker pricing and task design, most of the budget allocation techniques focus on homogeneous tasks. However, inherent heterogeneity in task batches can be exploited in order to customize crowd access according to the task types [152, 52]. We have also explored the applicability of the Access Path model for heterogeneous tasks in [125]. This recent work demonstrates that (i) the model can be extended to express the relationship

between access paths and task groupings, and (ii) the extended model can be further exploited to perform task-dependent budget allocation. However, the work also emphasizes that these extensions are beneficial only in the presence of heterogeneous task groups previously defined by the task designer.

In our work, we look at optimization techniques which aim at providing the best quality for a given budget constraint. Another interesting optimization problem is finding the least expensive crowd access strategy that satisfies a required quality constraint/target. A relevant work that tackles this problem is presented in [74]. The authors show that it is possible to ensure required correctness constraints with a probability $1 - \epsilon$, if the task redundancy is at least $O((K/q) \log(K/\epsilon))$, where K is the task cardinality and q defines the quality of the crowd equivalent to the probability of picking a non-spammer worker. The work however only applies to individual worker models and it would be interesting to understand how these results can be adapted to group-based models.

Another complementary line of work related to crowd access optimization is the design of efficient stopping conditions in crowdsourcing. Oftentimes, predefined quality constraints cannot be guaranteed due to task difficulty or high worker disagreement. As a result, it is beneficial to understand whether further crowd accesses can still improve the prediction quality. In [7], this problem is commonly solved by introducing stopping rules in the crowdsourcing process which adaptively estimate the additional benefit of collecting new labels.

Query optimization and access path selection. The crowd access optimization problem that we study in this thesis, was inspired by the use case of query optimization in crowdsourced databases. Crowdsourced databases extend the functionalities of conventional database systems to support crowd-like information sources. Quality assurance and crowd access optimization are envisioned as part of the query optimizer, which in this special case needs to estimate the query plans not only according to the cost but also to their accuracy and latency. Most of the previous work [50, 106, 122, 46] focuses on building declarative query languages with particular support for processing crowdsourced data. The proposed optimizers take care of (i) defining the order of execution of operators within query plans and (ii) mapping the crowdsourcable operators to micro-tasks, while the quality of the results is then ensured by requiring a minimum amount of responses. In our work, we propose a complementary approach by first ensuring the quality of each single database operator executed by the crowd.

Although the idea of access paths is crucial to query optimization for relational databases [136], in crowdsourced databases this abstraction is not fully explored. One of the few studies on this topic is presented in Deco [122]. Deco uses *fetch rules* to define how

data can be obtained either from humans or other external sources. In this regard, our *access path* concept is analogous to a *fetch rule* with an important distinction that an access path is associated with extra knowledge (error and cost) which enables the database optimizer to use them for quality assurance.

3.3 Problem definition: Access path selection

We now formulate the problem of crowd access optimization in the context of access path selection under budget constraints. Note that in this definition, we assume that the requester has already trained an Access Path model by learning its parameters from previous data. For cost optimization concerns, the requester is interested in optimally distributing the available budget across access paths to maximize the quality of new predictions from the model.

Problem 3.1 (ACCESS PATH SELECTION). *Given a task represented by a random variable Y , that can be solved by the crowd following N different access paths denoted with the random variables Z_1, \dots, Z_N , using a maximum budget B , the access path selection problem is to find the best possible access plan S_{best} that leads to a high-quality prediction of the outcome of Y .*

An access plan defines how many different people are chosen to complete the task from each access path. In Example 2.1, the access plan $S = [1, 2, 3]$ asks one pediatrician, two logopedists and three other parents. Each access plan is associated with a cost $c(S)$ and quality $q(S)$. For example, $c(S) = \sum_{i=1}^3 c_i \cdot S[i] = \80 , where c_i is the cost of getting one single answer through access path Z_i . In these terms, the access path selection problem can be generally formulated as:

$$S_{best} = \arg \max_{S \in \mathcal{S}} q(S) \text{ s.t. } \sum_{i=1}^N c_i \cdot S[i] \leq B \quad (3.1)$$

This knapsack maximization problem is tractable for modular objective functions (*e.g.* summing the quality of access plans) but is NP-Hard even for submodular functions [79]. Therefore, designing bounded and efficient approximation schemes is necessary to implement realistic crowd access optimization.

The choice of the quality objective function $q(S)$ is crucial to the problem. Since the crowd responses and the ground truth are not known ahead of time in crowdsourcing applications, the quality objective should evaluate the quality of predictions before the crowd access takes place. As explained in Section 3.5, we use *information gain* $IG(Y; S)$

3.3. Problem definition: Access path selection

as a quality objective $q(S)$ because it naturally combines access path accuracy and diversity [68, 85, 95]. Moreover, if applied in a suitably defined probabilistic model as the APM, information gain can explore diversity to provide realistic confidence values for predictions. These properties comply with the quality criteria that we define in Problem 2.1.

Motivation for Access Path Selection. Access path selection is essential for obtaining a good prediction as can be seen in the following example where different access plans provide different expected accuracy for the same budget. This example reflects our later experimental setup in the context of multiple predictions.

Example 3.1. *medicalQA* is a dataset of health-related questions (similar to the one in Example 2.1). Suppose the answer of each question is crowdsourced through two access paths Z_1 and Z_2 , e.g. a group of doctors and a group of patients. There are four task solvers available in each access path. For simplicity, let us also assume that the access paths have equal cost, $c_1 = c_2 = 1$. Table 3.1 shows a set of possible answers that are collected for each of the three questions currently in the dataset. Note that these answers and the ground truth are not known before requesting an answer on a certain access path but are shown here for illustration purposes only.

| Question | Z_1 | Z_2 |
|--------------------------|---------|---------|
| Q_1 (ground truth = 0) | 0 0 0 0 | 1 1 1 1 |
| Q_2 (ground truth = 0) | 1 1 1 0 | 0 0 0 1 |
| Q_3 (ground truth = 1) | 1 1 1 1 | 0 0 0 0 |

Table 3.1: Set of answers collected from two access paths for Example 3.1.

Imagine now that a user has a budget constraint $B = 3$ to solve each question and wants to do so for all questions in the dataset. There are three strategies that the user can follow for this purpose:

- S_1 : Three answers are randomly selected from Z_1 and Z_2 .
- S_2 : Three answers are randomly selected from Z_1 which is expected to be more accurate than Z_2 .
- S_3 : The answers are selected from a predefined combination of Z_1 and Z_2 . For illustration, we will look at the access plan $[2,1]$ that randomly selects two answers from the first access path and one from the second.

To compare these strategies, we calculate the expected accuracy for each strategy if majority vote is used for aggregation. Accuracy is defined as the percentage of questions

answered correctly and for this example it can take values from $\{1.0, 2/3, 1/3, 0.0\}$. Using the sample votes provided above, the probability of each strategy to correctly answer each of the questions is shown in Table 3.2. Next, Table 3.3 shows the overall expected accuracy and the detailed accuracy estimates per question and strategy.

| | | | |
|-------|-------|-------|-------|
| | S_1 | S_2 | S_3 |
| Q_1 | 1/2 | 1.0 | 1.0 |
| Q_2 | 1/2 | 0.0 | 3/8 |
| Q_3 | 1/2 | 1.0 | 1.0 |

Table 3.2: Probability of each strategy to correctly solve each of the questions.

| | | | | | |
|-------|-----------|-----------|-----------|-----------|-------------|
| | $a = 1.0$ | $a = 2/3$ | $a = 1/3$ | $a = 0.0$ | $E[S]$ |
| S_1 | 1/8 | 3/8 | 3/8 | 0.0 | 0.5 |
| S_2 | 0.0 | 1.0 | 0.0 | 0.0 | 0.67 |
| S_3 | 3/8 | 5/8 | 0.0 | 0.0 | 0.79 |

Table 3.3: Expected accuracy of each strategy.

As the example shows, access path selection is neither trivial nor simple. Here, asking three different workers on a single access path with a higher accuracy does not necessarily guarantee good result quality (see Q_2). Instead, investing part of the budget on workers from a different access path can effectively leverage diversity and improve output quality. We address crowd access optimization over different access paths in Section 3.5.

3.4 Information Gain as a quality measure

The first step of crowd access optimization is to estimate the quality of access plans before they are executed and then use this as an objective function. One attempt might be to quantify the *accuracy* of individual access paths in isolation, and choose an objective function that prefers the selection of more accurate access paths. However, due to statistical dependencies of responses within an access path (*e.g.*, correlated errors in the workers' responses), there is diminishing returns in repeatedly selecting a single access path. To counter this effect, an alternative would be to define the quality of an access plan as a measure of *diversity* (*e.g.*, as proposed in [66]). For example, we might want to prefer to equally distribute the budget across access paths. However, some

access paths may be very uninformative / inaccurate, and optimizing diversity alone might waste budget. Instead, we use the joint *information gain* $IG(Y; S)$ of the task variable Y in our model and an access plan S as a measurement of plan quality as well as an objective function for our optimization scheme. Formally, this is defined as:

$$IG(Y; S) = H(Y) - H(Y|S) \quad (3.2)$$

In Equation 3.2, an access plan S determines how many variables X to choose from each access path Z_i . $IG(Y; S)$ measures the reduction in entropy (as measure of uncertainty) of the task variable Y after an access plan S is observed. At the beginning, selecting from the most accurate access paths provides the highest uncertainty reduction. Nevertheless, if better access paths are relatively exhausted (*i.e.*, accessed relatively often), asking one more question in less accurate ones reduces the entropy more than continuing to ask on paths that were previously explored. This situation reflects the way how information gain explores diversity and increases the confidence of the prediction if evidence is retrieved from independent channels. Based on this analysis, information gain naturally trades accuracy and diversity. While plans with high information gain do exhibit diversity, this is only a means for achieving high predictive performance of the complete access plan. This measure is widely used in Bayesian experimental design aiming to optimally design experiments under uncertainty. In targeted crowdsourcing the concept has been recently applied [68, 95].

3.4.1 Information gain computation

The computation of the conditional entropy $H(Y|S)$ as part of information gain in Equation 3.2 is a difficult problem, as full calculation requires enumerating all possible instantiations of the plan with votes. Formally, the conditional entropy can be computed as:

$$H(Y|S) = \sum_{y \in Y, x \in X_S} p(x, y) \log \frac{p(x)}{p(x, y)} \quad (3.3)$$

X_S refers to all the possible assignments that votes can take according to plan S . We choose to follow the sampling approach presented in [85] which randomly generates samples satisfying the access plan according to our Bayesian Network model. The final conditional entropy will then be the average value of the conditional entropies of the generated samples. For a total number of G samples generated according to a plan S , the whole computation is performed as in Equation 3.4.

$$H(Y|S) = \frac{1}{G} \sum_{g=1}^G H(Y|x_g) = -\frac{1}{G} \sum_{g=1}^G \sum_{y \in Y} p(y|x_g) \log p(y|x_g) \quad (3.4)$$

This method is known to provide absolute error guarantees for any desired level of confidence if enough samples are generated. Moreover, it runs in polynomial time if sampling and the probabilistic inference from the network can also be done in polynomial time. Both conditions are satisfied by our model due to the tree-shaped configuration of the Bayesian Network. They also hold true for the Naïve Bayes baselines described in Section 2.5 as simpler tree versions the APM. Consequently, the overall computational cost of computing information gain on the APM is $\mathcal{O}(GNM)$. Generating a larger number of samples guarantees a better approximation.

3.4.2 Submodularity of information gain

Next, we derive the submodularity property of our objective function based on information gain in Equation 3.2. The property will then be leveraged by the greedy optimization scheme in proving constant factor approximation bounds. A submodular function is a function that satisfies the law of diminishing returns which means that the marginal gain of the function decreases while incrementally adding more elements to the input set.

Let \mathcal{V} be a finite set. A set function $F : 2^{\mathcal{V}} \rightarrow \mathbb{R}$ is submodular if $F(S \cup \{v\}) - F(S) \geq F(S' \cup \{v\}) - F(S')$ for all $S \subseteq S' \subseteq \mathcal{V}$, $v \notin S'$. For our model, this intuitively means that collecting a new vote from the crowd adds more information when few votes have been acquired rather than when many of them have already been collected. While information gain is non-decreasing and non-negative, it may not be submodular for a general Bayesian Network [85]. Information gain can be shown to be submodular for the Naïve Bayes Model for Access Paths (NBAP) in Figure 2.3 by applying the results from [85]. Here, we prove its submodularity property for the Bayesian Network of APM shown in Figure 2.4. Theorem 1 formally states the result and the full proof is given in Appendix A.1. Below we describe a short sketch of the proof.

Theorem 1. *The objective function based on information gain in Equation 3.2 for the Bayesian Network Model for Access Paths (APM) is submodular.*

Sketch of Theorem 1. In order to prove Theorem 1, we consider a generic Bayesian Network Model for Access Paths (APM) with N access paths, where each access path is associated with M possible votes from workers. To prove submodularity of the objective function, we consider two sets (plans) $S \subset S'$ where $S' = S \cup \{v_j\}$, *i.e.*, the plan S' containing one additional vote from access path j compared to plan S . Then, we consider adding a vote v_i from access path i and we prove the diminishing return property of adding v_i to S' compared to adding to S . The proof considers two cases.

ALGORITHM 1. GREEDY Crowd Access Optimization

```

1 Input: budget  $B$ 
2 Output: best plan  $S_{best}$ 
3 Initialization:  $S_{best} = \emptyset$ ,  $b = 0$ 
4 while  $(\exists i \text{ s.t. } b \leq c_i)$  do
5    $U_{best} = 0$ 
6   for  $i = 1$  to  $N$  do
7      $S_{pure} = \text{PurePlan}(i)$ 
8     if  $c_i \leq B - b$  then
9        $\Delta_{IG} = \text{IG}(Y; S_{best} \cup S_{pure}) - \text{IG}(Y, S_{best})$ 
10      if  $\frac{\Delta_{IG}}{c_i} > U_{best}$  then
11         $U_{best} = \frac{\Delta_{IG}}{c_i}$ 
12         $S_{max} = S_{best} \cup S_{pure}$ 
13       $S_{best} = S_{max}$ 
14       $b = \text{cost}(S_{best})$ 
15 return  $S_{best}$ 

```

When v_i and v_j belong to different access paths, *i.e.*, $i \neq j$, the proof follows by using the property of conditional independence of votes from different access paths given Y and using the “information never hurts” principle [85]. The case of v_i and v_j belonging to the same access path, *i.e.*, $i = j$, is technically more involved. For this case, we reduce the network to an equivalent network which contains only one access path Z_i and then use the “data processing inequality” principle [18, 31]. \square

This theoretical result is of generic interest for other applications and a step forward in proving the submodularity of information gain for more generic Bayesian networks.

3.5 Access path selection

3.5.1 Greedy approximation scheme

After having determined the joint information gain as an appropriate quality measure for a plan, the crowd access optimization problem is to compute:

$$S_{best} = \arg \max_{S \in \mathcal{S}} \text{IG}(Y; S) \text{ s.t. } \sum_{i=1}^N c_i \cdot S[i] \leq B \quad (3.5)$$

where \mathcal{S} is the set of all plans. An exhaustive search would consider $|\mathcal{S}| = \prod_{i=1}^N \frac{B}{c_i}$ plans out of which the ones that are not feasible have to be eliminated. Afterwards, from all the feasible plans, the one with the maximum information gain has to be selected. Nevertheless, efficient approximation schemes can be constructed given the similarity of the problem with analogous maximization problems for submodular functions under budget constraints [79].

Based on the submodular and non-decreasing properties of information gain we devise a greedy technique as illustrated in Algorithm 1 that incrementally finds a local approximation for the best plan. In each step, the algorithm evaluates the benefit-cost ratio U between the marginal information gain and cost for all possible access paths feasible to access. The marginal information gain is the improvement of information gain by adding to the current best plan one pure vote from one access path. Function $\text{PurePlan}(i)$ creates a plan that contains a single vote from the i -th access path. In the worst case, when all access paths have unit cost, the computational complexity of the algorithm is $\mathcal{O}(GN^2MB)$ which also includes the computation of information gain as in Section 3.4.1.

Except the greedy technique, we also studied a dynamic programming optimization scheme on the same objective. Although dynamic programming produces good approximate plans that are not statistically different from those selected by greedy optimization in practice, it does not improve the theoretical guarantees, and has a higher complexity.

3.5.2 Greedy approximation with varying costs

We now exploit the submodularity properties of the information gain in our considered Bayesian network model to prove theoretical bounds of the greedy optimization scheme. For the simple case of unit costs across all access paths, the greedy selection optimization scheme in Algorithm 1 guarantees a utility of at least $(1 - \frac{1}{e})$ ($= 0.63$) times the one obtained by optimal selection denoted by OPT [112]. This result is tight under reasonable complexity assumptions [48]. However, the greedy selection scheme fails to provide approximation guarantees for the general setting of varying costs of access paths [79]. By using techniques of partial enumeration combined with greedy selection or running multiple variants of the selection scheme can make it competitive [146, 85], however such an algorithm may not be practically useful and feasible because of overhead of running multiple iterations of the algorithm.

However, we exploit the following realistic property about the costs of the access paths and allocated budget to prove strong theoretical guarantees about our Algorithm 1. We

assume that the allocated budget is large enough compared to the costs of the access paths. Formally stating, we assume that the cost of any access path c_i is bounded away from total budget B by factor γ , *i.e.*, $c_i \leq \gamma \cdot B \forall i \in \{1, \dots, N\}$, where $\gamma \in (0, 1)$. We state the theoretical guarantees of the Algorithm 1 in Theorem 2 below. The complete proof is given in Appendix A.2.

Theorem 2. *The GREEDY optimization in Algorithm 1 achieves a utility of at least $\left(1 - \frac{1}{e^{(1-\gamma)}}\right)$ times that obtained by the optimal plan OPT. Hereby, γ denotes the maximal ratio of the cost of any access path $c_i \forall i \in \{1, \dots, N\}$ and the allocated budget B .*

For instance, for $\gamma = 0.5$, the Algorithm 1 achieves a utility of at least 0.39 times that obtained by OPT, while the approximation ratio for $\gamma = 0.10$ is 0.59.

Sketch of Theorem 2. We follow the structure of the proof from [79, 146]. The key idea is to use the fact that budget spent by the algorithm at the end of execution when it can not add an element to the solution is at least $(B - \max_{i \subseteq [1, \dots, N]} c_i)$, which is lower-bounded by $B(1 - \gamma)$. This lower bound on the spent budget, along with the fact that the elements are picked greedily at every iteration leads to the desired bounds. \square

These results are of practical importance in many other applications as the assumption of non-unit but bounded costs w.r.t. budget often holds in realistic settings.

3.6 Experimental Evaluation

In this set of experiments, we evaluate the efficiency of the proposed greedy approximation scheme to accurately choose plans of high quality that take diversity into account. For a fair comparison, we adapted the same scheme to the other baselines NBI (Naïve Bayes Individual in Section 2.4) and NBAP (Naïve Bayes for Access Paths in Section 2.5.2.1). We will use the following acronyms for the crowd access strategies: OPT (optimal selection), GREEDY (Greedy Approximation), RND (random selection), BEST (select votes from the best access path w.r.t. to accuracy), and EQUAL (select the same number of votes from each access path to increase diversity).

3.6.1 Greedy approximation and diversity management

The goal of this experiment is to answer the questions: “*How close is the greedy approximation to the optimal theoretical solution?*” and “*How does information gain as a*

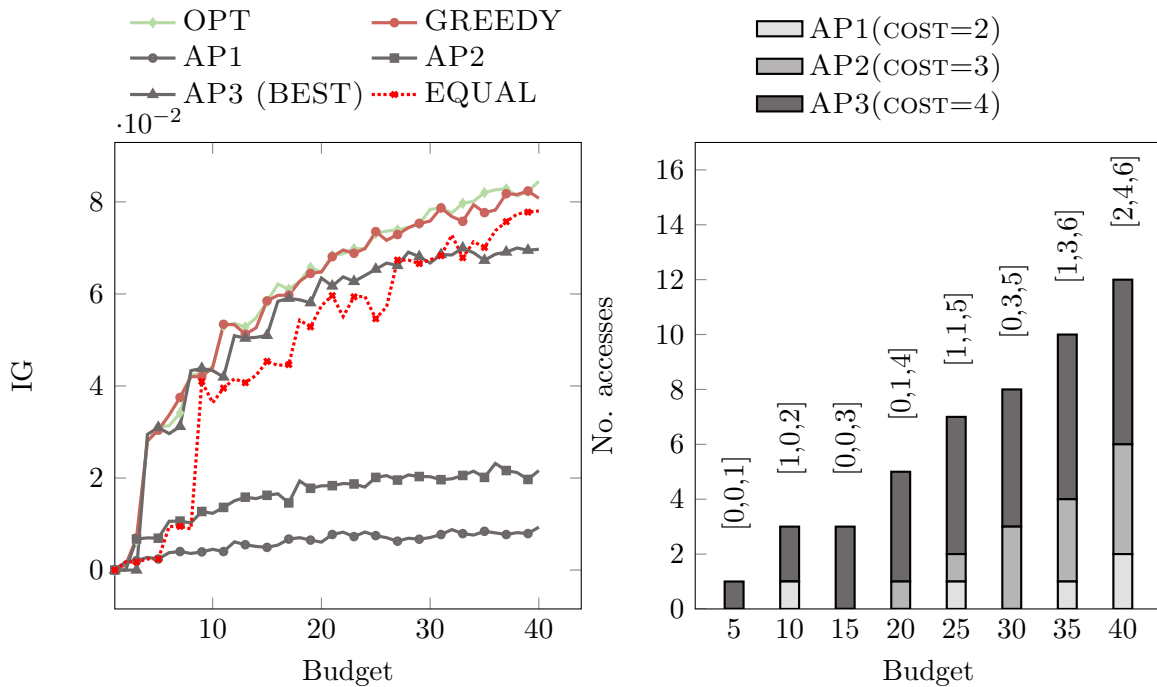


Figure 3.1: Information gain and budget distribution for ProbabilitySports (year=2002).

theoretical measure of uncertainty reduction exploit diversity? Figure 3.1 shows the development of information gain with varying budget for the optimal plan, the greedily approximated plan, the equal distribution plan, and three pure plans that take votes only from one access path. The quality of GREEDY is very close to the optimal plan. The third access path in ProbabilitySports (AP3, containing users with more than average relative score) reaches the highest information gain compared to the others. Nevertheless, its quality is saturated for higher budget values which encourages the optimization scheme to select other access paths as well. For example, if the budget constraint is $B = 40$, the optimal plan is [2,4,6]. For the same experiment, the NBAP model with the same optimization strategy selects the third access path only. Also, we notice that the EQUAL plan does not reach optimal values of information gain although it maximizes for diversity. This happens because part of the budget is misused while querying uninformative access paths. As we further show in the next experiments, the quality of predictions can be further improved if diversity is carefully planned by using information gain as an objective function. Note that the budget for experiments in this section does not always correspond to the number of votes because of the costs of the access paths are non-unit.

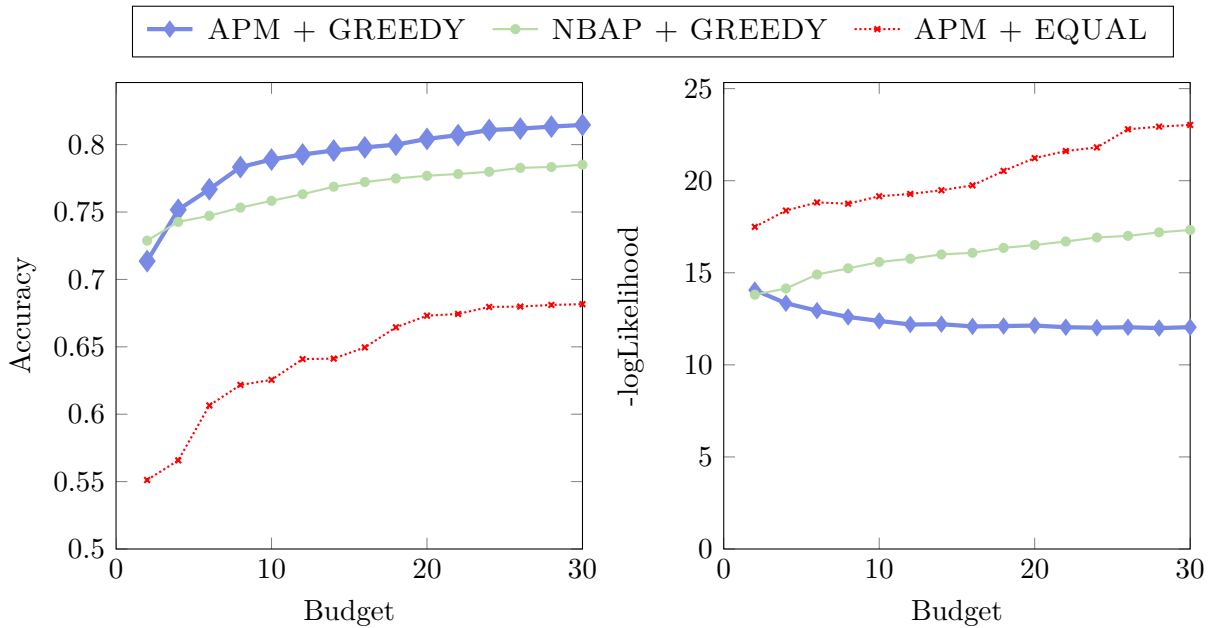


Figure 3.2: Crowd access optimization results for varying budget in CUB-200.

3.6.2 Optimization with budget constraints

This experiment combines together both the model and the optimization technique by following these four steps:

Step 1. Learn the model from historical training data.

Step 2. Find the best access plan using the learned model.

Step 3. Randomly choose votes from the testing data.

Step 4. Aggregate the chosen votes using the learned model.

The main question we want to answer here is: “*What is the practical benefit of greedy optimization on APM w.r.t. accuracy and negative log-likelihood?*” Figures 3.2 and 3.3 show results for CUB-200 and ProbabilitySports datasets. We discuss MedicalQA results in detail in the next experiment.

CUB-200.(Figure 3.2) For this dataset where the division of access paths is based on attributes, the discrepancy between NBAP and the APM is higher. As mentioned earlier, it is not possible to make a proper comparison with neither NBI nor MV as votes in the dataset do not directly answer on the final task. In practice, the search space of the optimization scheme in this data set is large due to the high number of access paths which results to many more feasible access plans. Since not all of the

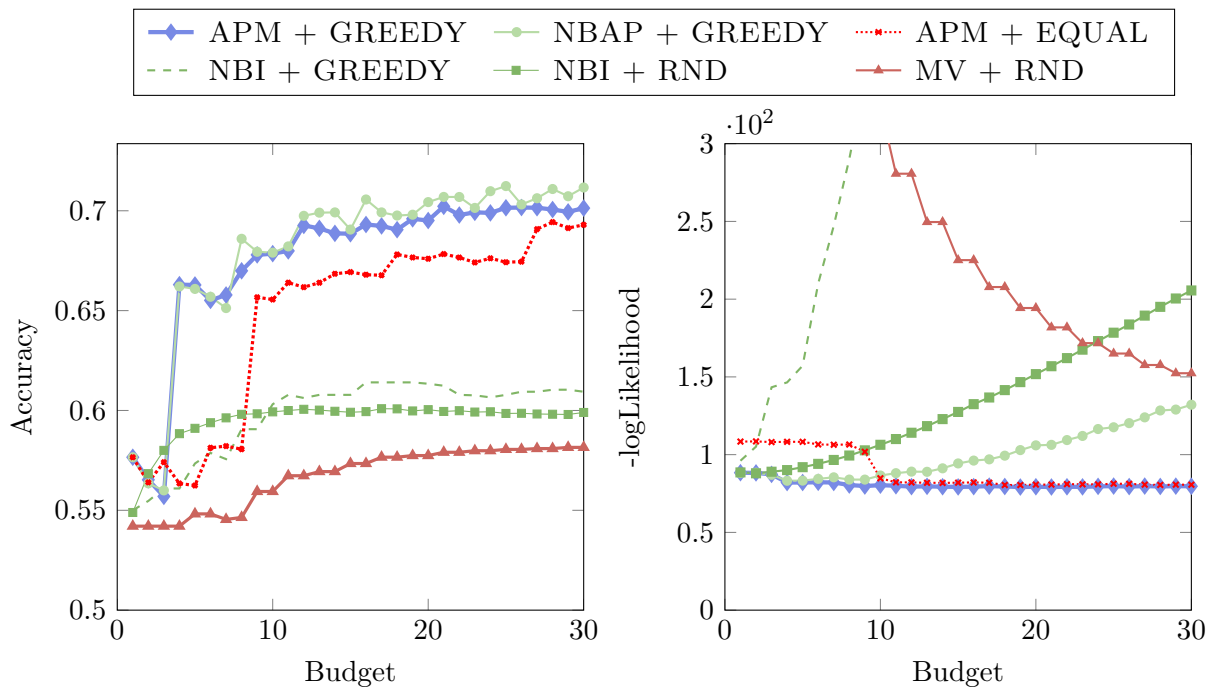


Figure 3.3: Crowd access optimization results for varying budget in *ProbabilitySports* (year = 2002).

attributes are informative for all tasks, EQUAL plans do not optimize for quality, their performance is significantly worse. We also plan to study the impact of feature selection for this specific setting.

ProbabilitySports.(Figure 3.3) Access Path based models (APM and NBAP) overperform MV and NBI. Since the plans for NBI target concrete users in the competition, the accuracy for budget values less than 10 is low as not all targeted users voted for all the events. Thus, we also present the performance of NBI with random access of votes (NBI+RND). In this dataset and configuration, access paths are inherently designed based on the accuracy of workers. This is why EQUAL plans do not offer a clear improvement while NBAP is advantaged in terms of accuracy by its preference to select the most accurate access paths only. APM is nevertheless more stable when both accuracy and negative log-likelihood are considered.

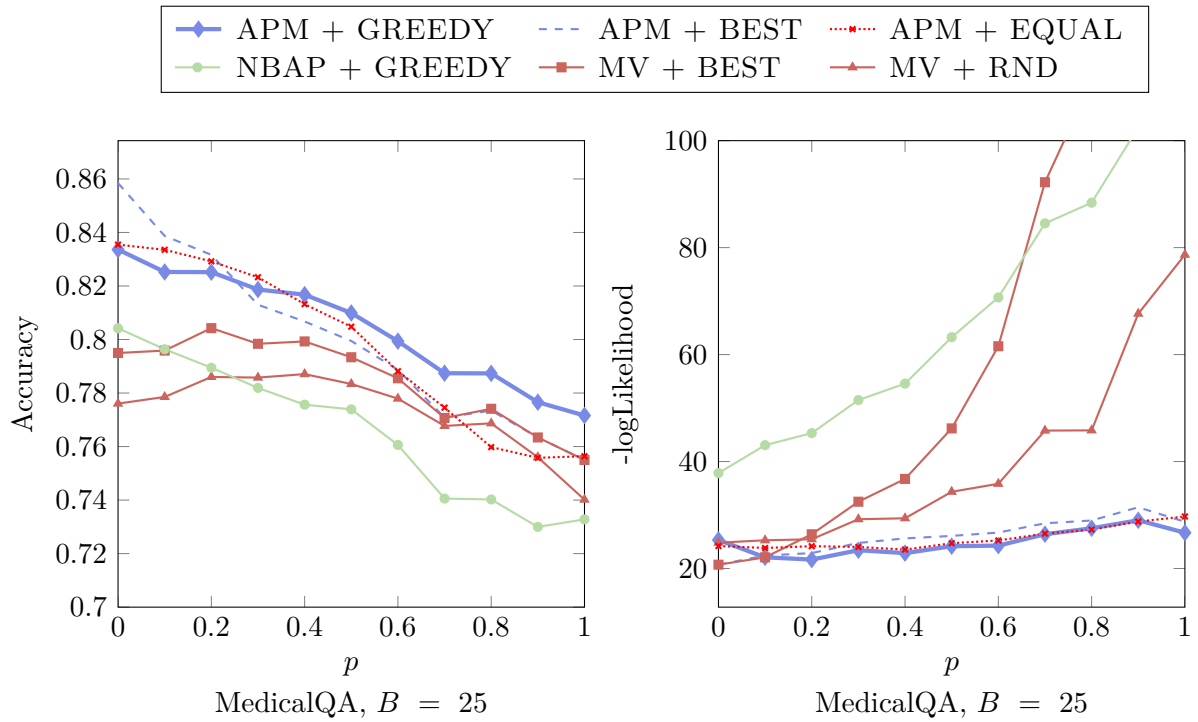


Figure 3.4: *Diversity impact on optimization. Varying access path correlation in MedicalQA.*

3.6.3 Impact of diversity

This experiment is designed to study the impact of diversity and conditional dependence on crowd access optimization and finally answer the question: “*How does greedy optimization on APM handle diversity?*”. One form of such dependency is within access path correlation. If this correlation holds, workers agree on the same answer. In contrast, if there is no dependency, the answer of a worker does not imply any information on the expected answer of another worker within the same access path. We experimented by varying the shared dependency within the access path as follows: Given a certain probability p , we decide whether a vote should follow the majority vote of existing answers in the same access path. For example, for $p = 0.4$, 40% of the votes will follow the majority vote decision of the previous workers and the other 60% will be withdrawn from the real crowd votes.

As shown in Figure 3.4, the quality of optimization drops for high dependency values but the Access Path Model is more robust to this. NBAP instead, due to overconfidence, accumulates all votes into a single access path which dramatically penalizes its

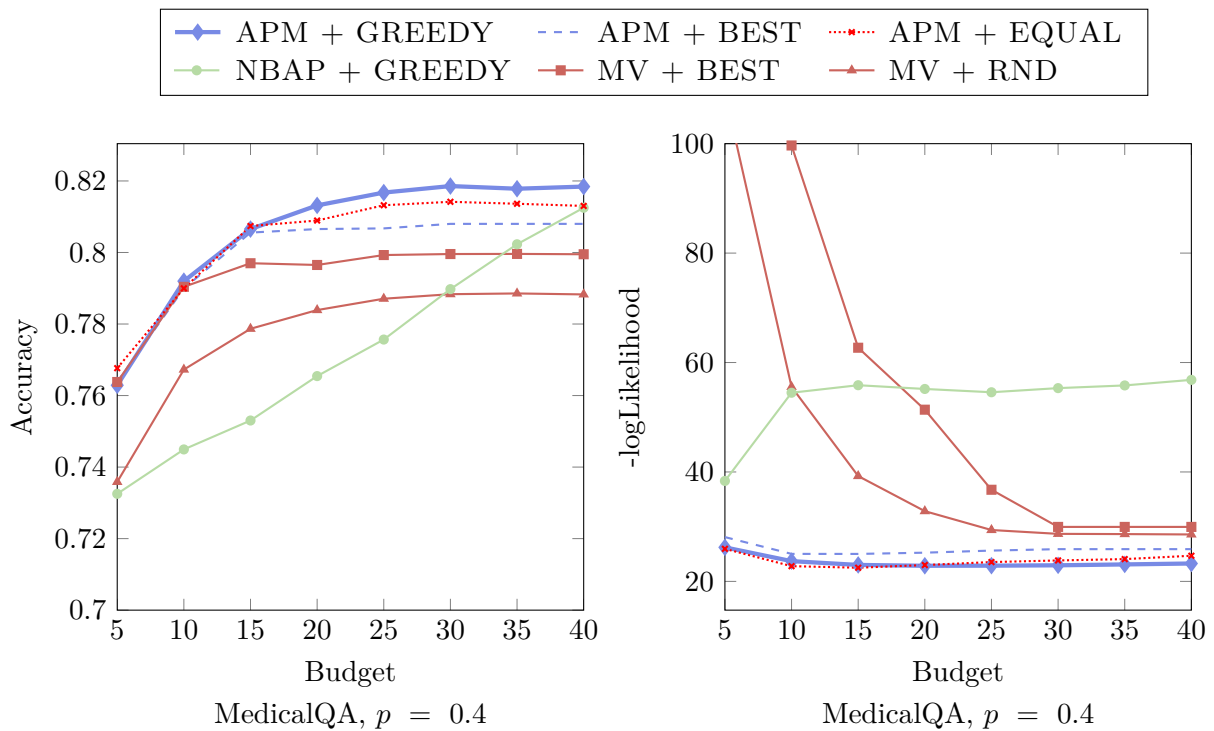


Figure 3.5: Diversity impact on optimization. Varying budget in MedicalQA.

quality. APM+BEST applies APM to votes selected from supposedly the access path with the best accuracy, in our case forums with doctors’ answers. Results show that for $p > 0.2$, it is preferable to not only select from the best access path but to distribute the budget according to the GREEDY scheme. It is interesting to see that due to high dependency, majority vote also becomes overconfident.

Furthermore, in Figure 3.5 we show results from the same experiment for $p = 0.4$ and varying budget. APM+GREEDY outperforms all other methods reaching a stable quality at $B = 30$. For stronger dependence values this stability is reached earlier which motivates the need to develop techniques that can dynamically stop the crowdsourcing process if no new insights are possible. EQUAL plans result to perform slightly worse than GREEDY *w.r.t.* accuracy although they are fairly applied over APM.

3.7 Summary

This chapter presented a crowd access optimization strategy specifically designed to collect new labels for making new predictions with the Access Path model. The strategy adapts the greedy approximation scheme for maximizing monotone and increasing submodular set functions. In this context, the scheme provides an approximate solution to the access path selection problem, which plans ahead how many different workers to access within the same access path. Due to the ability of the APM to model correlations within access paths, maximizing information gain for this model prevents redundant crowd accesses by exploring crowd diversity across access paths. Therefore, the resulting access plans are a mixed combination of various access paths according to the amount of information gain that they provide on the prediction.

While the Access Path model provides a practical tool for interpreting incoming crowd answers to make new predictions, the optimization strategy that we present here complements this tool for applications with budget limitations. In these applications, this strategy would guide crowd requesters to retrieve answers from the right set of crowd workers. Although the task allocation infrastructure in current crowdsourcing platforms is limited, we nevertheless applied these ideas in practice by implementing back-end libraries, which can perform online mapping of workers to tasks. Integrating such libraries within platforms would support requesters in seamless processes of crowd access optimization and label aggregation.

4

Budgeted Learning and Feature Selection

4.1 Overview

In previous chapters, we looked at quality control and optimization problems that raise while making new predictions from crowdsourced data. For this setting, we assumed that prediction models are trained with ample historical data. However, as the performance of machine learning algorithms crucially depends on the quality of the training sets, acquiring the most informative training data in a cost-effective way is fundamental to building good prediction models. In this chapter, we study feature-based classification problems for which the feature labels are unknown in the training phase, which is a commonly encountered setting in crowdsourcing [164]. The data acquisition algorithm can acquire the label of a desired feature for an instance and each such label has a fixed cost. While acquiring the most informative data for the problem, we consider two types of budget constraints, often present in crowdsourcing but also in other applications for medical diagnosis and sensor data aggregation.

Budget constraints at training phase Budget constraints at the training phase limit the total cost of training data acquisition by the algorithm. The end goal is to acquire the most informative labels to learn a classifier with low generalization error. This notion of budget constraint is different the one usually considered in active learning [137]. In active learning, the feature labels for all training in-

stances are known and the budget constraint bounds the cost of acquiring class labels. Also, the data acquisition is more fine-grained in our settings since the algorithm can pick any specific training instance and acquire only a desired set of features. In practical terms, budget constraints at the training phase map to the amount of budget that can be used in order to train a classifier before employing it to classify new instances.

Budget constraints at testing phase To perform classification at the testing phase, the algorithm again needs to acquire the feature labels for the test instance. This is often the case in many classification problems where the feature values need to be collected at a given cost for every new instance and can not be computed automatically. For example, in crowdsourcing prediction markets, medical diagnosis, and sensor data aggregation, new predictions in the testing phase are mostly based on newly acquired data for the specific instance that needs to be classified.

This naturally leads to a budget constraint at the testing phase, which in practice bounds the cost of making a prediction, often tackled implicitly via feature selection techniques. However, current techniques for budgeted learning and feature selection (as well as the respective budget constraints) have been studied and designed independently from each other. Several methods apply feature selection strategies only after the labels have been collected for all features during the training phase, which can be prohibitively expensive especially if the number of initial candidate features is high. Then, after learning the predictive parameters of all features, the least informative ones are discarded to reduce the cost of predictions at the testing phase [58, 57, 154]. Here, we consider both constraints simultaneously while learning a classifier.

The problem of learning with budget constraints at the training phase has been formerly studied in a noise-free setting for learning Naïve Bayes classifiers [41, 72, 103]. However, the proposed algorithms assume that the feature labels are noise-free, which is an unrealistic assumption for real-world applications. Hence, it is necessary to support more robust learning models for handling noise, which further exacerbates the challenges arising from the above mentioned budget constraints. To the best of our knowledge, this is the first work that considers both budget constraints while learning a classifier, in both the noise-free and noisy setting.

Figure 4.1 shows an example of making predictions from crowdsourced feature labels under budget constraints.

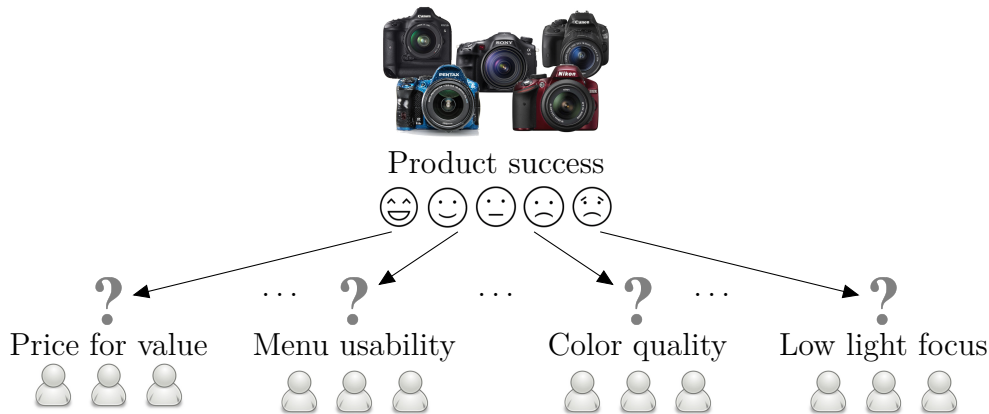


Figure 4.1: Example of feature-based crowdsourced predictions.

Example 4.1. FEATURE-BASED CROWDSOURCED PREDICTIONS

Imagine a company that produces various types of cameras and wants to implement a new strategy for predicting the overall success of its new products based on the customer evaluation of different product features: price for value, menu usability etc. The sales team collects this data via (paid) customer surveys on specific features. However, long surveys are cumbersome and time-consuming for the customers to complete. Moreover, some of the candidate features may be redundant / uninformative, and the evaluation from the customers may be noisy due to subjectivity or human error.

Given that there is no previous data available, the company: (i) has a limited budget for collecting data to train a model from its current products (i.e. training budget constraint), and (ii) prefers to have only a limited number of features in the surveys for the future products (i.e. testing budget constraint). Therefore, the questions of the sales team for this problem are:

Q1. What data should be collected in order to build a good training model?

Q2. Which features should be included in the final survey for predicting the customer satisfaction from new products.

Similar questions arise in numerous other problems of making predictions based on crowdsourced feature labels. Analogous applications include medical diagnosis and sensor data aggregation, where the budget constraints respectively relate to the cost of medical tests or sensor measurements.

We propose a novel learning algorithm, B-LEAFS, to *jointly* tackle the problems of **B**udgeted **L**earning and **F**eature **S**election for training and testing classifiers that are robust to noisy feature labels. B-LEAFS operates in a Bayesian framework, and main-

tains posterior distributions over all model parameters, thereby enabling us to capture the uncertainty in the model parameters about individual features. The algorithm makes greedy decisions for selecting the next feature label to acquire by exploiting the submodularity of information gain from a feature, conditioned on the current state of learning. In addition, it effectively balances exploration and exploitation by employing Thompson sampling [148].

4.2 Related work

Learning under budget constraints. Budgeted learning has previously been studied in the context of learning feature-based Naïve Bayes classifiers [41, 40, 103] under fixed budget constraints. The problem is however discussed orthogonal to feature selection and assumes accurate feature labels. [72] formally defines the joint problem but the proposed algorithms first learn the parameters of the whole candidate set and then adaptively select features via a bounded decision tree. While this adaptive approach improves the classification accuracy, it requires a significant amount of data for building a plausible decision tree. Recently, [89] studies the optimization of the computation time of feature extraction during training.

Another line of research is related to best-arm identification under budget constraints [100, 42]. Nevertheless, these methods are not designed for learning classifiers but rather for making a bounded selection of actions that maximize the joint reward. Bandit algorithms have been lately applied in the crowdsourcing setting [20, 177, 150, 151, 8] for worker selection purposes in the context of expert crowdsourcing. Differently from previous work on worker selection, the goal here is to select workers while learning about their expertise. Our work addresses similar exploration-exploitation trade-offs related to feature selection rather than worker selection for training and testing classifiers in a cost-efficient way. The main difference between the two problems is the fact that for the same example a worker can be accessed only once while a feature can be labeled multiple times from various workers. Both problems are however complementary to each other and integrating both into a single framework is an important avenue for future work.

Yet another optimization aspect relevant to crowdsourcing is related to applications where the task label is unknown both at training and testing time and is acquired at a given cost. This problem has been studied in recent work [98, 99] in the context of (re)active learning in crowdsourcing where the task labels are unknown but the feature labels are cost-free. Balancing this trade-off for the generic case of unknown

4.3. Problem definition: Learning and feature selection under budget constraints

task and feature labels would combine related aspects of active learning and budgeted learning. Although the key ideas of B-LEAFS (*i.e.* model sampling, credibility check, greedy feature selection) are valid even for these applications, the algorithm still needs to consider an additional cost-quality trade-off between labeling features for a new instance or labeling features for instances that are already in the dataset.

Learning from crowdsourced data. The rapid advances in crowdsourcing applications created new opportunities for the community to collect and annotate data. However, crowdsourcing processes for large-scale data acquisition are expensive and prone to noisy annotations. This motivated the need for applying active learning techniques [98, 110, 175] for reducing the cost of data collection. These approaches mainly focus on actively selecting tasks and workers and assume that feature labels are cost-free. The closest study to our method is presented by [134]. The authors propose a traditional Thompson Sampling algorithm for selecting observations from noisy sensors. As we show in the experimental section, the algorithm requires a longer exploration phase, which increases the cost of selecting the best features.

In this work, we use the Access Path model for learning from crowdsourced data. Similar approaches group workers [156] and tasks [71] to overcome the challenges of data sparsity and task-dependent biases. Furthermore, other worker selection and crowd access optimization strategies [59, 73] can be leveraged to achieve a fine-grained optimization customized to the pool of crowd workers.

Finally, recent work has focused on crowdsourcing feature engineering and discovery for machine learning classifiers [178, 29, 143]. Our ideas are complementary to feature discovery as they can be applied to refine the learning model after obtaining a viable candidate feature set from the crowd.

4.3 Problem definition: Learning and feature selection under budget constraints

The problem of budgeted learning and feature selection is tied to the specific classification model that needs to be trained. The goal is to learn a good classifier on a given budget. This section introduces the two models based on which we build our algorithm: Naïve Bayes model (Section 2.4), and Access Path model (Section 2.5) as a representative of models that handle crowdsourcing errors. Later, we formally define our problem in the context of these two models.

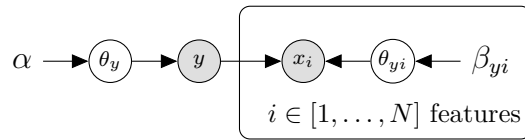


Figure 4.2: *The Naïve Bayes model — plate representation.*

4.3.1 Feature-based classification models

Naïve Bayes model. Figure 4.2 shows the plate representation of a Naïve Bayes classifier. Note that this is the same model as the one we use in Chapters 2 and 3, shown in a graphical representation in Figure 2.2. In this chapter, we make use of the plate representation in order to explain the details of our algorithm.

Each training instance in the Naïve Bayes model is characterized by a class / task variable Y taking values in the domain \mathcal{Y} , and a set of N feature variables $X = \{X_1, \dots, X_N\}$. The Naïve Bayes model considers a noise-free setting, where each feature label is observed exactly once and its value is considered to be correct. For further details on learning and inference for Naïve Bayes, we refer the reader to Section 2.4.

θ_y and θ_{yi} denote the underlying probability distributions for the task variable and the feature variables respectively. In this work, we will deal with categorical tasks and features. Therefore, θ_y and θ_{yi} are both modeled as Dirichlet distributions, where α and β_{yi} are the corresponding hyperparameters. We assume uniform hyperparameters on all the variables. The goal of training in this model is to learn the conditional probabilities of each feature given the task, *i.e.* $p(X_i|Y = y, \beta_{yi}, D)$ where D is the set of feature labels that have been collected so far. These probabilities correspond to the maximum a posteriori (MAP) estimates [104] and can be computed exactly when Y is also fully observable (*i.e.* there exists a ground truth for the final task). Otherwise, alternative expectation-maximization (EM) techniques [37] can be employed to estimate parameters. Predictions are then performed via Bayesian Inference.

The main assumption in this model is the conditional independence of the feature variables given the task. The assumption may not always hold true due to possible correlations between features. This can lead to Naïve Bayes predictions being overconfident when features are not carefully engineered beforehand which further motivates the need for feature selection.

Access Path model. As mentioned earlier, the Naïve Bayes model handles a single label per feature for a given instance which is considered to be correct. In crowdsourcing settings where the labels acquired from the workers may be noisy, it is common to ask

4.3. Problem definition: Learning and feature selection under budget constraints

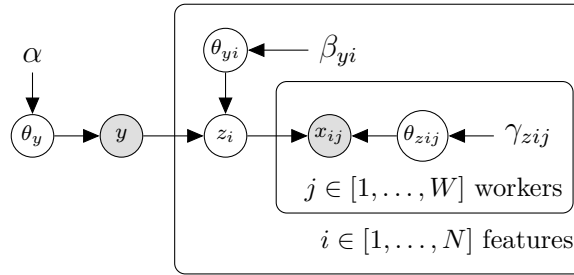


Figure 4.3: *The Access Path model — plate representation.*

the same question / label to multiple workers to ensure quality. In order to support such noisy labels, we adapt the Access Path model as shown in Figure 4.3 — a natural extension to Naïve Bayes while being able to handle multiple (possibly noisy) labels per feature. This design was originally proposed to represent groups of correlated worker answers (*i.e.* access paths), with the goal of optimizing the prediction costs. In our work, we leverage this model by adapting the notion of worker groups to represent classification features. For further details on learning and inference for the Access Path model, we refer the reader to Section 2.5.

The main benefit of this model is that it decouples the noise / information that inherently comes with the features (*i.e.* $p(Z_i|Y = y, \beta_{yi}, D)$) from the noise / information that is further introduced by the workers (*i.e.* $p(X_{ij}|Z_i = z, \gamma_{zij}, D)$) by introducing a middle layer of latent variables Z_i for $i \in 1 \dots N$. Similar to Dawid and Skene (2015), learning is performed via an expectation-maximization (EM) algorithm [37], while inference during the prediction step marginalizes over the latent feature variables. Due to the high sparsity of worker participation, this model shares the parameters for the workers within the same feature, which was shown to improve the accuracy of predictions. We follow the same guideline in our experiments as our algorithm runs in training time where the data sparsity is even higher. Nevertheless, if it is known that some of the workers have a sufficient participation to accurately estimate their quality, the model could be further extended to differentiate such workers from the rest by not sharing the θ_{zij} parameters.

4.3.2 Problem statement

We study feature-based classification problems for which the task label $Y = y$ is known during the training phase, and needs to be predicted by the classifier at the testing phase. The feature labels are unknown for both the training and testing instances and can be acquired at a given cost. In practice, there are various problems for which this

assumption holds. In Example 4.1 for instance, the company may already know the success rate of previous products but it may not know the values of individual features and which ones are most informative for predicting the success of new products. Another example is animal species recognition in images as we show in our experiments. For this problem, one can bootstrap a classifier with training examples for which the species in the image is already known but the visual feature labels need to be determined.

Consider a Naïve Bayes model with N conditionally independent feature variables $X = \{X_1, \dots, X_N\}$ as presented in Figure 4.2. At the beginning, the classifier has no knowledge about the underlying parameters of θ_{yi} , and we initialize them from the hyperparameters β_{yi} . At a given time, a new feature label is acquired ($Y = y, X_i = x_i$) for the feature X_i on a task with label $Y = y$. Based on this labeled instance, the algorithm can update the distributions θ_{yi} . For illustration purposes, let us consider the following example: Suppose X_i is a feature variable that takes values in the domain $\{a, b, c\}$ while the task variable Y takes binary values $\{yes, no\}$. The current state of the model has $\theta_{yes,i} \sim Dir(3, 2, 5)$. After observing a new labeled instance ($y = yes, X_i = b$) the posterior distribution will shift to the new state $\theta_{yes,i} \sim Dir(3, 3, 5)$.

In the context of the Access Path model shown in Figure 4.3, the algorithm acquires labels to learn the distributions of both the features (θ_{yi}) and the crowd workers (θ_{zij}) at the same time. As it is usually the case in crowdsourcing, on each feature observation we ask a total number of W different crowd workers. This will lead to a cost reduction of W from the available budget B .

Budget B at the training phase First, the goal of *budgeted learning* is to learn the model parameters during the training phase under a given budget constraint B , with the goal of minimizing the classification error during the testing phase. The budget here limits the number of acquired labels represented by the labeled set D such that $|D| \leq B$. In crowdsourcing applications, this maps to the number of times the learner accesses the crowd. In Example 4.1, B is the number of times the algorithm asks a customer to evaluate a camera feature. While the learning problem is well understood for infinite budget, the budgeted version is known to be NP-HARD even for simplified variants when a feature is allowed to be queried only once [103].

Budget K at the testing phase Second, the goal of *feature selection* is to select a set of the best K features for classification in the testing phase where $K \leq N$. Hence, K is a budget constraint applicable to the testing phase. In Example 4.1,

K corresponds to the number of camera features that will be included in the final survey design. For the Naïve Bayes model, the feature selection problem is also NP-HARD [48], with viable approximation guarantees possible through greedy selection approaches [86, 146].

In our final problem definition below, we aim at tackling both challenges while learning a classifier, in both the noise-free and noisy setting which correspond to the two presented models above.

Problem 4.1 (BUDGETED LEARNING AND FEATURE SELECTION). *Given a feature-based classification task Y that can be solved via a set of N candidate features $X = \{X_1, \dots, X_N\}$ with unknown labels that can be acquired through crowdsourcing, the goal is to learn a classification model θ under a budget constraint B that can make high-quality predictions of Y using only $K < N$ most informative features.*

4.4 Budgeted Learning and Feature Selection

In this section, we first describe current approaches on our two-fold problem, and then we present B-LEAFS as an algorithm for budgeted learning and feature selection in the context of the Naïve Bayes and the Access Path model.

4.4.1 Existing approaches

The purpose of any budget allocation strategy for training is to produce a classifier that in the future is going to make good predictions. The trivial approach would be to uniformly allocate the budget among all features without distinguishing the good features from the bad ones. This is also known as the ROUNDROBIN algorithm, and as shown in our experimental evaluation and previous work, is generally inferior to other schemes as some of the features are more informative than others. Other strategies continue allocating budget to the same feature as long as this reduces the loss on the task variable (*i.e.*, the conditional entropy $H(Y|X_i)$) after this i -th feature is observed in isolation to the others. BIASEDROBIN [103] is one representative algorithm in this spectrum and was shown to be competitive [41] when there exists only a relatively small number of candidate features. However, for a larger number of candidates this algorithm (i) fails to deal with correlated features, and (ii) delays the selection of the best candidates. These issues are then handled by another group of GREEDY selection methods which select in each iteration the i -th feature that reduces the overall

loss of the whole model when added to the currently selected feature set X_S (*i.e.*, the conditional entropy $H(Y|X_{S \cup \{i\}})$). These methods are known to be efficient with strong theoretical guarantees [85] for fixed and known feature parameters learned from sufficient historical data. Our work is inspired from similar greedy selection techniques but adapted for the case of unknown feature parameters that are updated as the data is collected. A closely related approach, referred to as TSGREEDY in our experiments, proposed in [134], employs Thompson Sampling by greedily selecting in each iteration a whole set of K features from a sampled model. As discussed further during our experimental evaluation, this approach has a high exploration cost when feature labels are noisy and is applicable only to the Naïve Bayes model.

For further information, we provide a detailed description of these algorithms in Section 4.5.1.

4.4.2 Limitations and challenges

The fundamental issue with the aforementioned techniques is related to the fact that the decision-making on which feature to observe next is based on the point estimates of the feature parameters for the feature variables X_i . If the collected data for the feature is not sufficient, these estimates might not be good representatives of the underlying parameter distribution. If so, the observed value of the feature may be far away from its expected value which can then lead to selecting non-informative features and making wrong classifications in the testing phase.

4.4.3 B-Leafs for Naïve Bayes

The key idea of our approach is to use the posterior distributions over the parameters (*e.g.* θ_y and θ_{y_i} for Naïve Bayes) in order to encode the uncertainty about the true parameter value. Based on this Bayesian framework, we design the B-LEAFS algorithm inspired by Thompson Sampling [148, 9], which is a natural way to balance exploration and exploitation for comparing various probability distributions. Next, we describe how we apply these ideas for Naïve Bayes and the Access Path model.

The B-LEAFS strategy (Algorithm 2) performs one iteration per feature selection / search (Lines 10-19). The k -th iteration is allowed to spend a maximum budget of $b_k = \frac{B}{2^k}$ when $k - 1$ number of features have already been added to the best set. The reasoning behind this allocation is based on the experimental observation that the early iterations of the algorithm explore more and require more budget than the later ones.

4.4. Budgeted Learning and Feature Selection

ALGORITHM 2. B-LEAFS

```

1 Input:    feature variables  $X = \{X_1, \dots, X_N\}$ 
2           training budget constraint  $B$ 
3           testing constraint  $K$  features
4           credibility constraint  $\delta$ 
5 Output:  parameters  $\tilde{\theta} = \{\tilde{\theta}_y, \tilde{\theta}_{y_i} \forall y \in \mathcal{Y}, i \in [1, \dots, N]\}$ 
6           best feature set  $S$  s.t.  $|S| = K$ 
7 Initialize:  $S = \emptyset, D = \emptyset$ 
8           uniform priors  $\alpha, \beta = \{\beta_{y_i} \forall y \in \mathcal{Y}, i \in [1, \dots, N]\}$ 
9 for ( $k = 1; k \leq K; k++$ ) do
10    $b_k = \lfloor \frac{B}{2^k} \rfloor$  //  $\frac{B}{2^{k-1}}$  for  $k = K$ 
11    $i^* = \text{null}$ 
12   while ( $(\neg \text{ISCREDIBLE}(i^*, D, \beta, \delta))$  and  $(b_k > 0)$ ) do
13      $\tilde{\theta} = \text{SAMPLEMODEL}(D, \alpha, \beta)$ 
14      $i^* = \arg \max_{i \in [1, \dots, N] \setminus S} \text{IG}(Y; X_{S \cup \{i\}} | \tilde{\theta})$ 
15     Draw  $y \sim P(Y)$ 
16     Observe  $x_{i^*}$  for a task s.t.  $Y = y$ 
17      $D = D \cup \{(y, x_{i^*})\}$ 
18      $b_k = b_k - 1, B = B - 1$ 
19    $S = S \cup \{i^*\}$ 
20  $\tilde{\theta} = \text{MAP}(D, \alpha, \beta)$ 
21 return  $\tilde{\theta}, S$ 

```

ALGORITHM 3. SAMPLEMODEL - Naïve Bayes

```

1 Input:    collected data  $D$ , priors  $\alpha, \beta$ 
2 Output:  parameters  $\tilde{\theta} = \{\tilde{\theta}_y, \tilde{\theta}_{y_i} \forall y \in \mathcal{Y}, i \in [1, \dots, N]\}$ 
3 Sample  $\tilde{\theta} \sim P(\theta | D, \alpha, \beta)$ 
4 return  $\tilde{\theta}$ 

```

This iteration-based constraint ensures that each feature search does not spend more than a maximum amount of budget before including a new item in the best set even if the feature is not sufficiently credible. Every iteration performs the following steps:

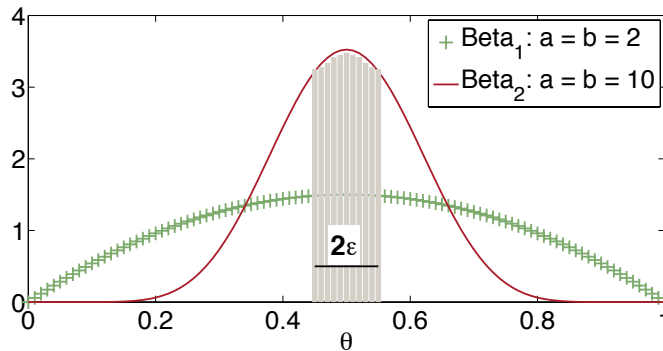


Figure 4.4: Example of binary parameter distributions

4.4.3.1 Model sampling

(Line 13) On each feature label acquisition, B-LEAFS samples a set of parameters from the current posterior parameter distribution θ . Model sampling is essential for our method as it helps to balance exploration vs. exploitation trade-offs. The more data we observe from a certain feature, the more likely it is for the sampled parameter to be close to its mean value. Consequently, the early decisions of B-LEAFS will tend to explore more features while the later decisions will mostly focus on exploiting the current best ones. This is the reason why the initial iterations require more budget than the others. For Naïve Bayes, sampling is trivial and is performed as in Algorithm 3. First, we compute the Dirichlet posterior distributions θ from the feature value counts in the current dataset D , and then sample a model $\tilde{\theta}$ from the resulting distributions.

4.4.3.2 Submodular feature selection

(Line 14) Similar to the Thompson Sampling algorithm, we decide on which features to observe next based on the sampled model parameters $\tilde{\theta}$. More precisely, we select the feature i^* that brings the most information in addition to what the current best set S already provides, which prevents the algorithm from including correlated features. Note that we use information gain as a decision-theoretic objective function which has been shown to be submodular for both the models described in the previous section due to the feature conditional independence assumption. The benefit of using a submodular set function is that it enables the application of efficient greedy approximation schemes [146]. Moreover, information gain can be approximately computed via sampling from the model network as shown in [85].

4.4.3.3 Parameter credibility check

(Line 12) Before adding a feature to the best set, B-LEAFS checks whether: (i) enough budget has been invested for the feature search, and (ii) the posterior distributions of this feature, θ_{yi^*} , are sufficiently credible and concentrated around the mean. For instance, imagine two binary features whose conditional distribution $P(X_i|Y)$ follows a Bernoulli distribution. Suppose also that the conjugate priors for these features are Beta distributions as shown in Figure 4.4. Both these distributions have a mean value of $\mu = 0.5$. However, Beta_1 has only a few observations and the mass of distribution within a credible interval $[\mu - \epsilon; \mu + \epsilon]$ as shown in the shaded area is smaller than for Beta_2 . Formally, we define the credibility of a feature X_i to be within a 2ϵ credibility interval around its mean as follows:

$$q_i = \sum_{y \in Y} \frac{P(Y = y)}{|\mathcal{X}_i|} \sum_{x \in \mathcal{X}_i} \int_{\mu - \epsilon}^{\mu + \epsilon} \text{Beta}(a_{yi}^x, a_{yi}^0 - a_{yi}^x) d\theta_{yi}^x \quad (4.1)$$

Here, a_{yi}^x is the number of times value $x \in \mathcal{X}_i$ has been observed when $Y = y$, and a_{yi}^0 is the total number of examples for which $Y = y$. Note that a_{yi}^x represents concentration parameters of the Dirichlet distribution for θ_{yi} . The normalization by $\frac{P(Y=y)}{|\mathcal{X}_i|}$ ensures that the credibility of the whole feature is weighed according to the distribution of Y and the feature variable cardinality. The function ISCREDBLE returns true if $q_i \geq 1 - \delta$, and false otherwise. The function is generic and can use other notions of parameter concentration such as variance.

In our implementation, we fix $\epsilon = 0.05$ while varying the credibility parameter δ according to the budget constraints. Our initial guideline with this respect is that the algorithm should spend more than $\frac{B}{N}$ budget units on the selected features which is the amount of budget that ROUNDROBIN would use. Without making any assumptions on the conditional distribution of $P(X_i|Y)$, one can compute a corresponding δ_L value for this guideline. Furthermore, one can also compute a corresponding δ_U value for the ideal case when the algorithm would spend on average a budget of $\frac{B}{K}$ on each selected feature. The closer δ is to δ_U , the tighter the credibility requirements are. As the algorithm also requires budget for exploration, δ_U is an unrealistic requirement. In our experiments, we observe that $\delta = \frac{\delta_U - \delta_L}{2}$ is sufficiently stringent to detect informative features.

4.4.3.4 Discussion

The B-LEAFS algorithm is based on a generic class of greedy selection algorithms for maximizing monotone submodular set functions [146, 48]. The crucial difference here

| k | #observations per feature | Selected features (S) | | | | | | | | |
|-----|-----------------------------------------------------------------------------------------------------------------------------|---------------------------|----|----|----|----|----|----|----|--------------------------|
| 1 | <table border="1"><tr><td>4</td><td>5</td><td>7</td><td>5</td><td>3</td><td>6</td><td>2</td><td>28</td></tr></table> | 4 | 5 | 7 | 5 | 3 | 6 | 2 | 28 | $\{X_8\}$ |
| 4 | 5 | 7 | 5 | 3 | 6 | 2 | 28 | | | |
| 2 | <table border="1"><tr><td>20</td><td>34</td><td>14</td><td>14</td><td>13</td><td>10</td><td>12</td><td>28</td></tr></table> | 20 | 34 | 14 | 14 | 13 | 10 | 12 | 28 | $\{X_2, X_8\}$ |
| 20 | 34 | 14 | 14 | 13 | 10 | 12 | 28 | | | |
| 3 | <table border="1"><tr><td>36</td><td>34</td><td>16</td><td>19</td><td>16</td><td>13</td><td>15</td><td>28</td></tr></table> | 36 | 34 | 16 | 19 | 16 | 13 | 15 | 28 | $\{X_1, X_2, X_8\}$ |
| 36 | 34 | 16 | 19 | 16 | 13 | 15 | 28 | | | |
| 4 | <table border="1"><tr><td>36</td><td>34</td><td>19</td><td>32</td><td>20</td><td>13</td><td>18</td><td>28</td></tr></table> | 36 | 34 | 19 | 32 | 20 | 13 | 18 | 28 | $\{X_1, X_2, X_4, X_8\}$ |
| 36 | 34 | 19 | 32 | 20 | 13 | 18 | 28 | | | |

Table 4.1: Example of running B-LEAFS on the *nursery* dataset for $B = 200$ and $K = 4$.

is that the objective function (*i.e.* information gain in our case) is computed on the sampled model for balancing exploration and exploitation. The data collection then (Line 17) follows a traditional greedy selection, while the feature selection (Line 19) instead appends new features only when they are sufficiently credible, which is the major distinction of B-LEAFS from the traditional Thompson Sampling algorithm. Both decisions are guided by the marginal increase of information gain after observing one additional feature value.

Example 4.2. In Table 4.1 we show an example of running B-LEAFS on the *nursery* dataset from the UCI repository [96]. In this example, we required the algorithm to spend a maximum budget of $B = 200$ while selecting $K = 4$ (out of 8) features. The optimal set of features without training phase budget constraints is $\{X_1, X_2, X_4, X_8\}$. Each element in the collected data vector represents the number of times a feature has been observed.

The algorithm adds one feature per iteration (marked in gray) in the set S and most of the budget is spent in the first two iterations of the algorithm. After the last iteration, the algorithm manages to recover the optimal set of features $\{X_1, X_2, X_4, X_8\}$ which is the set of features that would have been selected from a model trained with all the available data in the dataset. Note that the amount of data assigned to the algorithm in this case ($B = 200$) is only 2% of the overall data available in *nursery*.

B-LEAFS does not spend more budget on a feature once the feature has been added to S . One could observe that the final model θ could still benefit from reaccessing such features for further improving their parameters. However, once the best set of features is known, such refinements can be achieved in a more cost-efficient way by periodically retraining the model with new instances in the testing phase.

ALGORITHM 4. SAMPLEMODEL - Access Path model

```

1 Input:    collected data  $D$ , priors  $\alpha, \beta, \gamma$ 
2 Output:  parameters  $\tilde{\theta} = \{\tilde{\theta}_y, \tilde{\theta}_{yi}, \tilde{\theta}_{zij} \forall y, z, i, j\}$ 
3 for ( $t = 1; t \leq T; t++$ ) do
4    $\theta_y^{(t)} \sim P(\theta_y^{(t-1)} | \alpha, D)$ 
5   for ( $i = 1; i \leq N; i++$ ) do
6     foreach  $d \in D$  do
7        $z_i^{(t)}[d] \sim P(Z_i | \theta_{yi}^{(t-1)}, D)$  //update  $Z_i$  for instance  $d$ 
8        $\theta_{yi}^{(t)} \sim P(\theta_{yi} | z_i^{(t)}[1 \dots |D|], \beta_{yi}, D)$ 
9       for ( $j = 1; j \leq W; j++$ ) do
10       $\theta_{zij}^{(t)} \sim P(\theta_{zij} | z_i^{(t)}[1 \dots |D|], \gamma_{zij}, D)$ 
11 return  $\theta^{(t)}$ 

```

4.4.4 B-Leafs for the Access Path model

B-LEAFS for the Access Path model follows the same structure as Algorithm 2. However, due to the introduction of the feature layer with latent variables Z_i , model sampling and the parameter credibility check are accordingly adjusted.

4.4.4.1 Model sampling

Sampling from the Access Path model parameters entails sampling from θ_y , as well as θ_{yi} and θ_{zij} . In contrast to the Naïve Bayes model, here the actual counts that involve the Z_i variables cannot be observed. We overcome this problem by applying Gibbs sampling [51] as a Markov Chain Monte Carlo algorithm for computing approximate observations from a joint distribution of random variables. Gibbs sampling is a suitable choice for the Access Path model given that the hidden variables θ_y , θ_{yi} , and Z_i are characterized by conditional probability distributions as depicted in Figure 4.3.

As shown in Algorithm 4, we implement these ideas for alternatively sampling: (i) the value of the latent feature variables Z_i (Line 7), and (ii) the model parameters θ_{yi} and θ_{zij} (Line 8 and 10). Each iteration samples one latent variable at a time given the state of the rest of the other variables present in the model. Note that the Z_i feature variables need to be sampled for all the data that has been collected so far which requires multiple passes through the data. However, for more practical applications,

the algorithm can still benefit from parallelizing sampling across features thanks to the conditional independence assumption.

4.4.4.2 Parameter credibility check

Similar to what we discussed earlier, B-LEAFS checks the credibility of the feature parameters by using the definition in Equation 4.1. For the Access Path Model we ensure that both the feature and crowd parameters (θ_{yi} and θ_{zij}) satisfy this condition. Depending on the amount of crowdsourcing redundancy W and noise, one may also enforce different δ -credibility requirements for θ_{yi} and θ_{zij} . For simplicity, in all our experiments we apply the same δ to both the parameters.

4.5 Experimental evaluation

Datasets. In this section, we show experimental results on two types of open and publicly available data sources: datasets from the UCI Machine Learning Repository [96] and the CUB-200 birds classification dataset [163]. Both datasets consist of categorical features and tasks. UCI datasets are labeled by domain experts and contain a single expert label per feature.

CUB-200 instead, was created as part of a large-scale crowdsourced data collection for bird species recognition. The dataset contains 6,033 images allocated over 200 different species. There are 288 candidate binary features in total, and the authors collected 5-10 crowdsourcing labels per feature per image. Therefore, for this task workers solved tasks with image-related questions like: “*Is the color of the bird’s beak yellow?*”. The collected answers then are used as labels for the corresponding features. The dataset does not have any ground truth on the true feature values. However, one can measure the amount of disagreement which is on average 3%-10% per image. Some features have a higher disagreement than others which can reach up to 50% for features that are hard to distinguish.

4.5.1 Baselines

We compare our approach with the following three different baselines from related work:

RoundRobin. [103] This algorithm uniformly allocates the budget across features in a round-robin fashion, applying thereby the pure-exploration strategy as in Algo-

ALGORITHM 5. ROUNDROBIN

```

1 Input:    feature variables  $X = \{X_1, \dots, X_N\}$ 
2           training budget constraint  $B$ 
3           testing constraint  $K$  features
4 Output: parameters  $\tilde{\theta} = \{\tilde{\theta}_y, \tilde{\theta}_{y_i} \forall y \in \mathcal{Y}, i \in [1, \dots, N]\}$ 
5           best feature set  $S$  s.t.  $|S| = K$ 
6 Initialize:  $S = \emptyset, D = \emptyset$ 
7           uniform priors  $\alpha, \beta = \{\beta_{y_i} \forall y \in \mathcal{Y}, i \in [1, \dots, N]\}$ 
8  $i = 1$ 
9 while ( $B \geq 0$ ) do
10   Draw  $y \sim P(Y)$ 
11   Observe  $x_i$  for a task s.t.  $Y = y$ 
12    $D = D \cup \{(y, x_i)\}$ 
13    $B = B - 1$ 
14    $i = i \bmod N + 1$ 
15  $\tilde{\theta} = \text{MAP}(D, \alpha, \beta)$ 
16  $S = \text{GREEDY}(\tilde{\theta}, K)$ 
17 return  $\tilde{\theta}, S$ 

```

rithm 5. Note that none of the algorithms we describe here involves explicit feature selection. For a fair comparison, we first learn a model from the collected data and then run a submodular GREEDY feature selection scheme on the resulting model [85]. This selection corresponds to Line 16 for ROUNDROBIN. In every step, the greedy algorithm selects the feature i that maximizes the marginal increase in information gain $\text{IG}(Y; X_{S \cup \{i\}})$ after adding this feature to the current best set. This algorithm is explained in detail in Section 3.5 for the Access Path model.

BiasedRobin. [103] As shown in Algorithm 6, this strategy continues observing the same feature as long as this action reduces the loss on the task variable, *e.g.* the conditional entropy $H(Y|X_i)$. Note that this notion of loss does not take into account sets of features but only the current feature in isolation. This may lead to selecting redundant features that are correlated with each other. Once the expected value of the loss does not decrease anymore, the algorithm starts exploring other features. Based on this design, the approach tends to quickly exploit features that seem to be more informative without exploring the whole candidate set. However, this optimistic behavior may prevent the algorithm from exploiting features that have not been encountered yet before the budget is exhausted.

ALGORITHM 6. BIASEDROBIN

```

1 Input:    feature variables  $X = \{X_1, \dots, X_N\}$ 
2           training budget constraint  $B$ 
3           testing constraint  $K$  features
4 Output: parameters  $\tilde{\theta} = \{\tilde{\theta}_y, \tilde{\theta}_{y_i} \forall y \in \mathcal{Y}, i \in [1, \dots, N]\}$ 
5           best feature set  $S$  s.t.  $|S| = K$ 
6 Initialize:  $S = \emptyset, D = \emptyset$ 
7           uniform priors  $\alpha, \beta = \{\beta_{y_i} \forall y \in \mathcal{Y}, i \in [1, \dots, N]\}$ 
8  $i = 1$ 
9  $\tilde{\theta} = \text{MAP}(D, \alpha, \beta)$ 
10  $l_{old} = \text{IG}(Y; X|\tilde{\theta})$ 
11 while ( $B \geq 0$ ) do
12   Draw  $y \sim P(Y)$ 
13   Observe  $x_i$  for a task s.t.  $Y = y$ 
14    $D = D \cup \{(y, x_i)\}$ 
15    $B = B - 1$ 
16    $\tilde{\theta} = \text{MAP}(D, \alpha, \beta)$ 
17    $l_{new} = \text{IG}(Y; X_i|\tilde{\theta})$ 
18   if ( $l_{new} \leq l_{old}$ ) then
19      $i = i \bmod N + 1$ 
20  $S = \text{GREEDY}(\tilde{\theta}, K)$ 
21 return  $\tilde{\theta}, S$ 

```

TsGreedy. [134] is the closest approach to our work although it is applicable only for the Naïve Bayes model. It applies a traditional Thompson Sampling algorithm as shown in Algorithm 7, which runs $T = \frac{B}{K}$ iterations. Depending on the Each iteration involves three main steps: (1) model sampling as in Line 9 (2) submodular selection of K features via the GREEDY algorithm as in Line 11(3) observing the K features and updating the current model as in Line 12. In contrast to our approach, this algorithm selects K features at a time and does not check the credibility of feature parameters before including them in the best set. As a result, the algorithm has a higher exploration cost especially in the presence of noise. For comparison purposes, we adapted TSGREEDY for the Access Path model by employing the same Gibbs Sampling algorithm as in Algorithm 4.

For all experiments, we keep 80% of the data for training and 20% for testing. All

ALGORITHM 7. TSGREEDY

```

1 Input:    feature variables  $X = \{X_1, \dots, X_N\}$ 
2           training budget constraint  $B$ 
3           testing constraint  $K$  features
4 Output: parameters  $\tilde{\theta} = \{\tilde{\theta}_y, \tilde{\theta}_{y_i} \forall y \in \mathcal{Y}, i \in [1, \dots, N]\}$ 
5           best feature set  $S$  s.t.  $|S| = K$ 
6 Initialize:  $S = \emptyset, D = \emptyset$ 
7           uniform priors  $\alpha, \beta = \{\beta_{y_i} \forall y \in \mathcal{Y}, i \in [1, \dots, N]\}$ 
8 while ( $B \geq 0$ ) do
9    $\tilde{\theta} = \text{SAMPLEMODEL}(D, \alpha, \beta)$ 
10   $k = \min(B, K)$ 
11   $S = \text{GREEDY}(\tilde{\theta}, k)$ 
12  for ( $i \in S$ ) do
13    Draw  $y \sim P(Y)$ 
14    Observe  $x_i$  for a task s.t.  $Y = y$ 
15     $D = D \cup \{(y, x_i)\}$ 
16   $B = B - k$ 
17   $\tilde{\theta} = \text{MAP}(D, \alpha, \beta)$ 
18   $S = \text{GREEDY}(\tilde{\theta}, K)$ 
19 return  $\tilde{\theta}, S$ 

```

methods select feature labels from the training set and the resulting model with only the selected features is then evaluated on the testing set. This process is repeated 16 times for each experiment.

Example 4.3. *To illustrate the differences between the different approaches, in Table 4.2 we show an example of running all approaches on the breast-cancer UCI dataset by assigning a budget constraint of $B = 100$ and $K = 2$. The optimal set of features for this dataset is $\{X_2, X_3\}$ and results to a 0.04 prediction error. Although none of the methods is able to recover the full best set, the differences between the various algorithms can be observed in the final set of selected features S and the amount of budget that is spent to learn each of these features in the collected data vector. BIASEDROBIN misses the selection of X_2 as it quickly decides to exploit the previous feature. TSGREEDY is able to identify X_2 as one of the best features but does not spend sufficient budget on it which leads to a higher error. In fact, the budget in TSGREEDY is more uniformly distributed. B-LEAFS instead manages to have a lower error rate given that*

| Algorithm | #observations per feature | Selected features (S) | Error |
|-------------|----------------------------|---------------------------|-------|
| ROUNDROBIN | 12 11 11 11 11 11 11 11 11 | $\{X_2, X_6\}$ | 0.10 |
| BIASEDROBIN | 18 10 7 16 6 15 7 18 3 | $\{X_1, X_8\}$ | 0.11 |
| TSGREEDY | 6 17 13 12 2 15 4 16 15 | $\{X_2, X_8\}$ | 0.08 |
| B-LEAFS | 1 45 10 0 2 38 2 1 0 | $\{X_2, X_6\}$ | 0.05 |

Table 4.2: Example of running all the algorithms on the *breast-cancer* dataset for $B = 100$ and $K = 2$.

it is able to better refine the parameters of the features in S by investing more budget in them and spending significantly less budget in less informative features (e.g. X_9 is the least informative feature here).

4.5.2 Evaluation on the Naïve Bayes model

Prediction error for varying budget constraints in training phase. We first show results on the Naïve Bayes model. In Figure 4.5 we show the error rate of the resulting classifiers in testing phase on two different UCI datasets while varying the training budget constraint B . The *nursery* dataset has $N = 8$ categorical features while the *breast-cancer* dataset has $N = 10$ categorical features. B-LEAFS is able to make more accurate predictions at a lower cost. More interestingly, for the *breast-cancer* dataset it is able to identify the top most informative features even though the best features of this dataset are comparably informative. BIASEDROBIN instead quickly overexploits some features and fails to select the best set. Finally, TSGREEDY is slower in identifying the best features due its longer explorative behavior.

Prediction error for varying budget constraints in testing phase. In the experiment in Figure 4.6, we vary the testing phase constraint K , *i.e.* the number of features that can be used for prediction. In the *nursery* dataset, only two of the features are informative for classification which explains why the error improvement saturates for all algorithms when $K > 2$. In general, we also observe that our approach (as well as BIASEDROBIN and TSGREEDY) are more beneficial when it is required to select a small number of features. Otherwise, if K is comparable to the size of the candidate set N , their performance converges to the simplistic ROUNDROBIN.

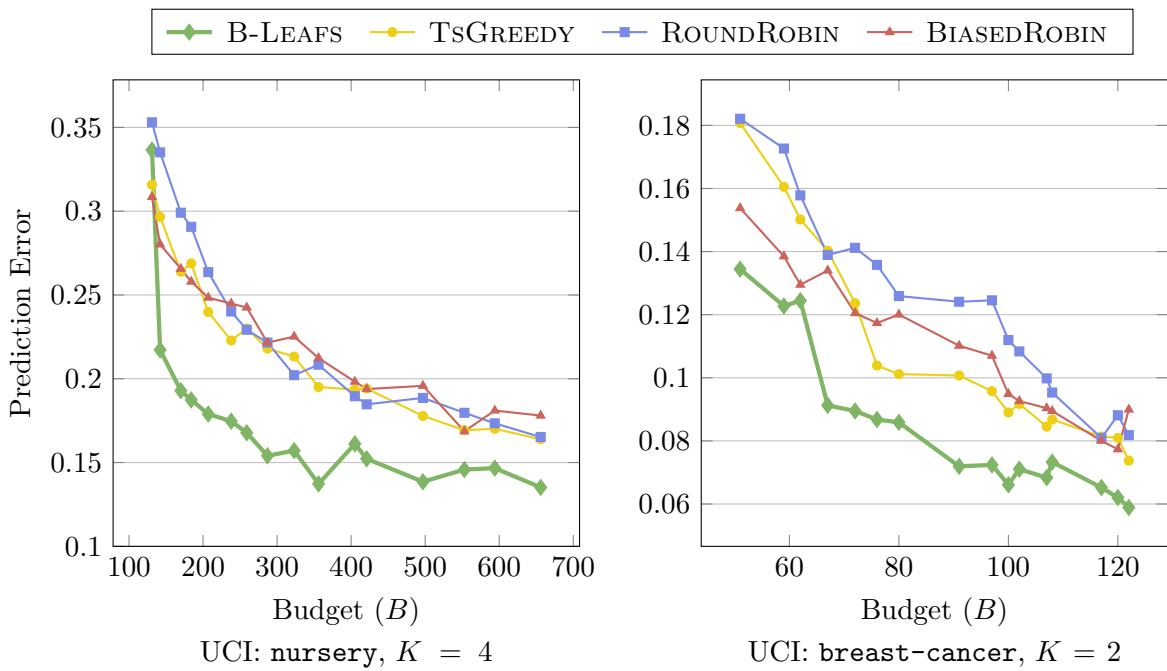


Figure 4.5: Learning and feature selection on Naïve Bayes. Varying B .

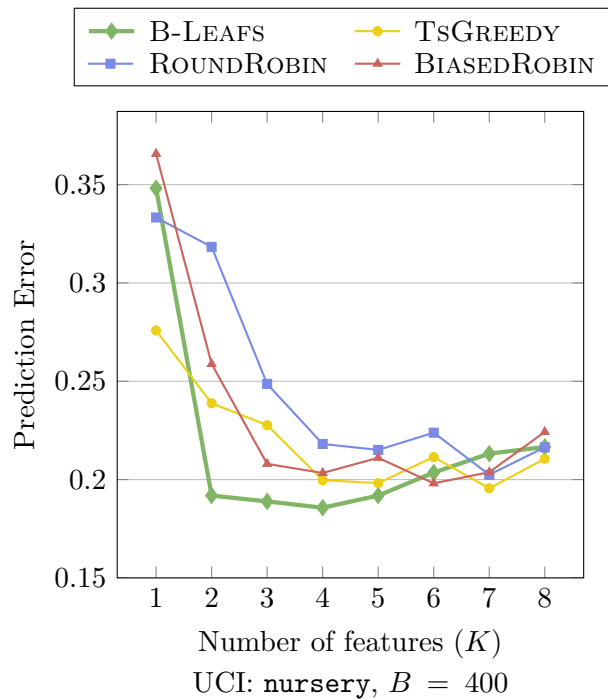


Figure 4.6: Learning and feature selection on Naïve Bayes. Varying K .

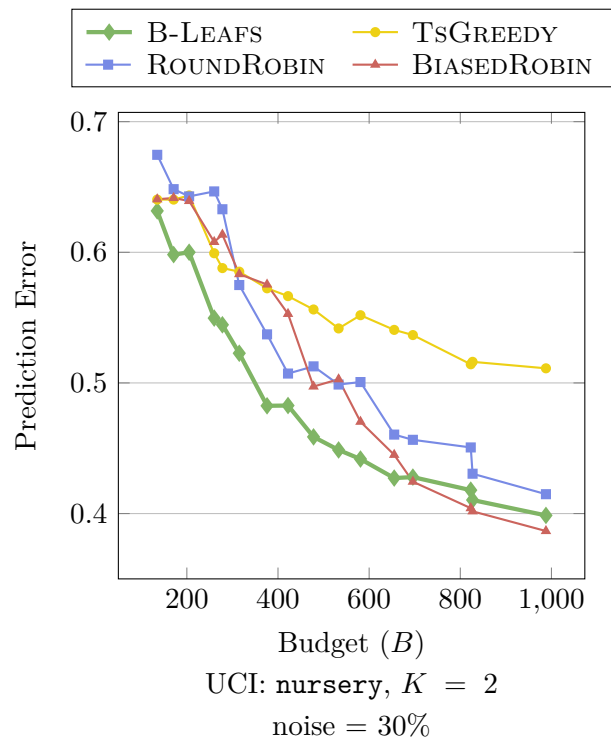


Figure 4.7: Learning and feature selection on the Access Path model. Synthetic noisy crowd.

4.5.3 Evaluation on the Access Path model

Synthetic noisy crowd. To evaluate our approach in a crowdsourcing setting, we initially add synthetic noise of 30% to the UCI datasets. In the experiment shown in Figure 4.7, for instance, we synthetically generate 5 feature labels based on the actual feature label in the dataset. In 70% of the cases, the label will correspond to the true value. The rest of the labels, are uniformly picked from the rest of the possible feature values. Multiclass datasets with non-binary features like `nursery` are highly sensitive to such noise. The interesting observation here is that, `TsGREEDY`'s performance significantly deteriorates in noisy settings, although it is fairly comparable to other baselines for noise-free observations. `B-LEAFS` then is more beneficial for lower rather than higher training budget constraints.

Real-word crowd. Figures 4.8 shows results on the CUB-200 datasets, on two different bird species. Note that due to the lack of ground truth for feature values, we can only measure the average worker disagreement on all features. Obviously, some features have a higher disagreement than others (up to 50%). These experiments involve one-vs-all classification tasks for birds that belong to the same category (*e.g.* different

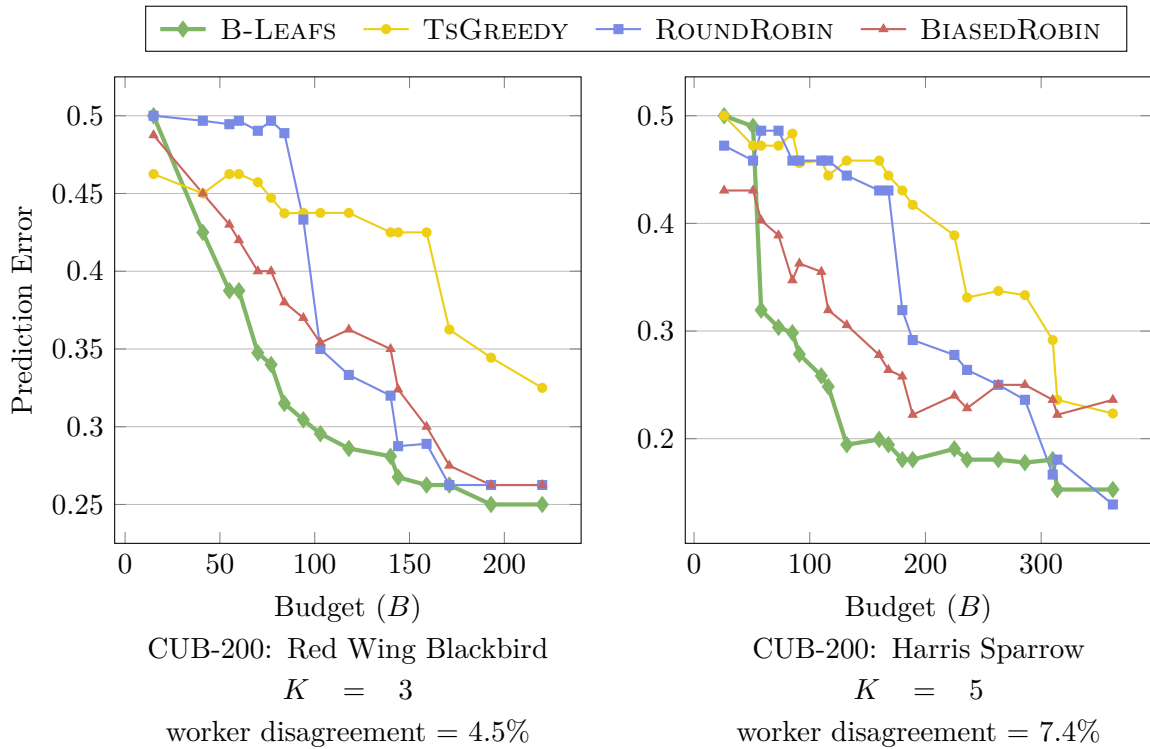


Figure 4.8: Learning and feature selection on the Access Path model. Real-world crowd.

kinds of sparrows) which is a more challenging task than classifying birds that belong to different categories. For each species, we include in the candidate feature set the top K most informative features and $N - K$ other randomly selected ones ($N = 20$). Results show that B-LEAFS outperforms the prediction error of BIASEDROBIN. All strategies are comparable for high budget except TSGREEDY, which again has a long exploration phase.

4.5.4 Noise impact

Finally, in the experiments in Figures 4.9 and 4.10, we study the behavior of both learning models under two different noise regimes: *uniform* noise and *biased* noise. For both regimes we synthetically generate 5 feature labels for both the training and the testing splits of the dataset. In the uniform noise case, the label either replicates the true feature value or is uniformly picked with a varying probability (horizontal axis) from the possible feature values different from the true value. This simulates random but non-biased worker mistakes. The biased noise regime instead imitates

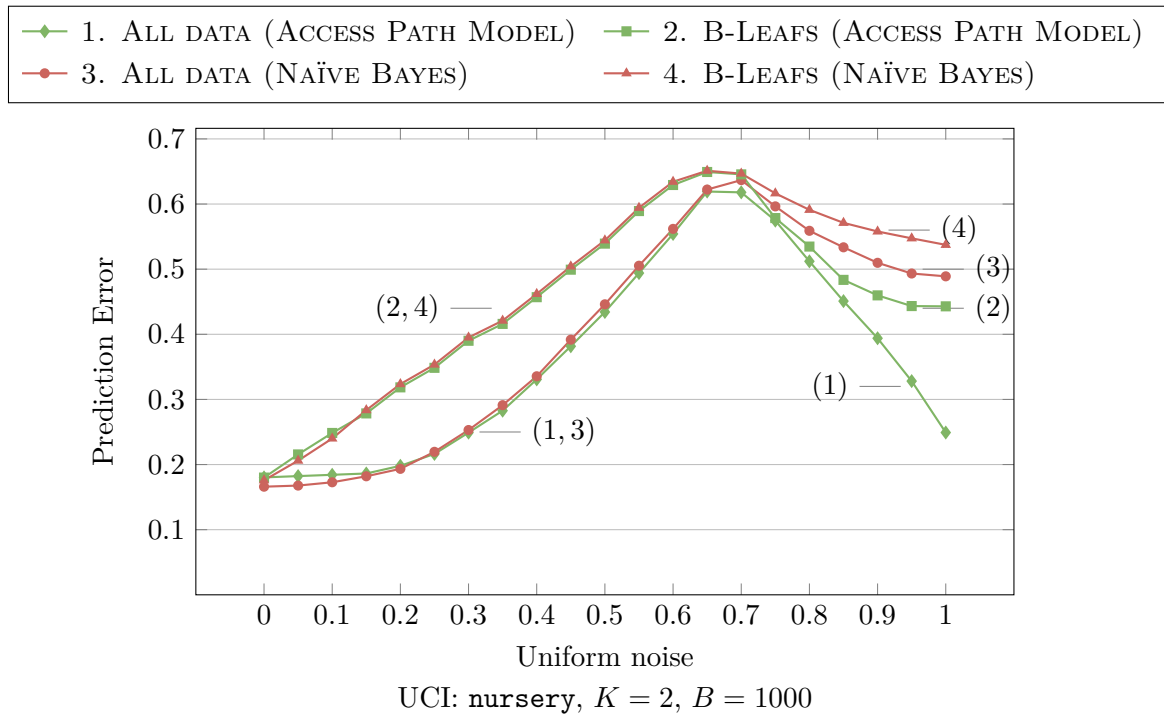


Figure 4.9: Uniform noise impact on the prediction error of models build from the data collected from B-LEAFS and the whole dataset.

situations when workers could have a strong bias in consistently confusing the true value for another erroneous feature value. Note that, both noise regimes are present in real-world crowdsourcing platforms. Moreover, in these experiments we also show the discrepancy of the prediction error of models trained with all the available data in the dataset (ALL DATA) and the models trained with only the data collected by B-LEAFS with a a budget constraint of $B = 1000$. In both settings, we select $K = 2$ best features.

Uniform noise. Figure 4.9 shows that in the uniform noise regime both learning models have comparable accuracy when noise is lower than 0.65. However, for higher levels of noise, the Access Path model is able to better recover from the uniform noise mistakes by inversely interpreting the feature labels. Also, the B-LEAFS algorithm is able to construct a better model if applied together with the Access Path model for noisy feature labels.

Biased noise. Similar observations are also depicted in Figure 4.10 for biased noise. However, these results also show that biased mistakes are more difficult to recover as the biased noise may lead to the selection of a non-optimal set of features. This explains

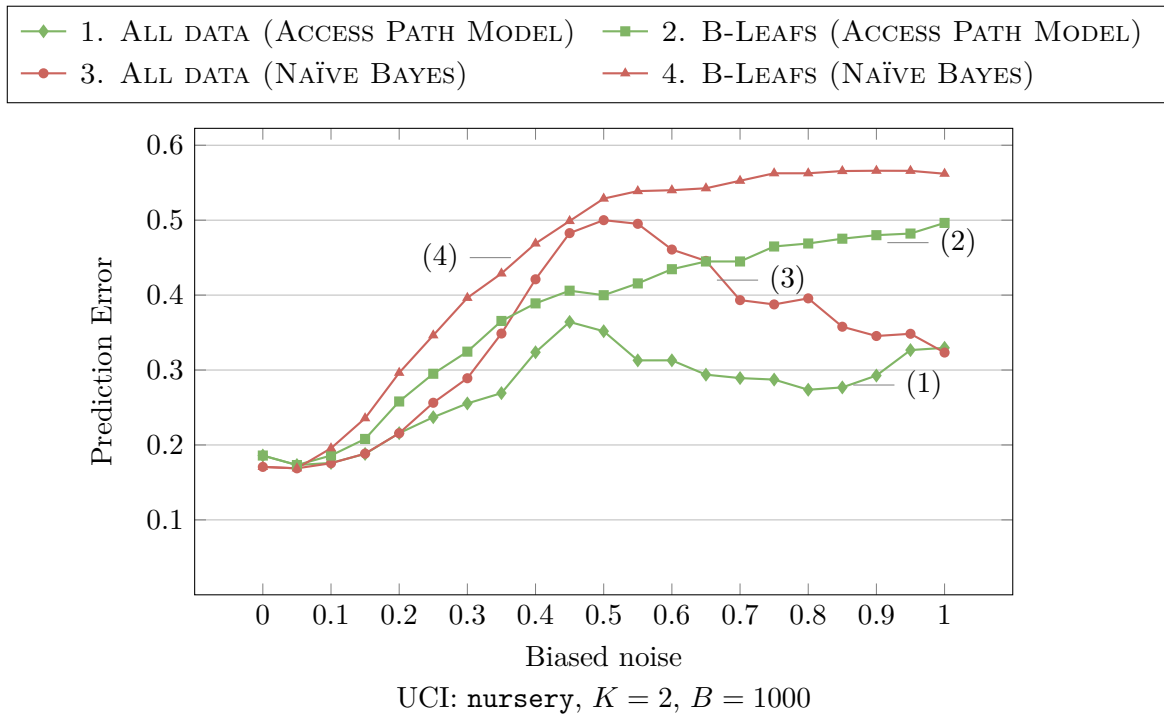


Figure 4.10: *Biased noise impact on the prediction error of models build from the data collected from B-LEAFS and the whole dataset.*

the discrepancy between the models built from B-LEAFS and the ALL DATA setting. These results highlight the importance of modelling and handling such biases in early stages of model training in order to ensure the selection of the right set of features.

4.6 Summary

In this chapter, we studied the problem of building machine learning models from crowdsourced data under training and testing budget constraints. In particular, we focused on feature-based classification models and proposed a budgeted learning and feature selection algorithm, B-LEAFS, which naturally balances exploration-exploitation trade-offs. This approach adaptively acquires crowdsourced feature labels for learning classifiers that can make accurate predictions at a lower cost.

Our experimental evaluation shows that budgeted learning and selection techniques are most efficient for restricted budget constraints: low training budget or low number of features. For ample training budget or a high number of features, traditional uniform budget allocation techniques are comparably efficient. In addition, these results also confirm that biased noise regimes, where workers consistently mistake one feature value for another, are more challenging to handle as they can lead to the selection of a non-optimal set of features. This motivates the need for employing prediction models that can appropriately encode such biases. We used the Access Path model as a representative of such models and as a natural extension of our work presented in Chapters 2 and 3. Applying B-LEAFS to other probabilistical models presented in previous work requires the adaptation of model sampling and greedy feature selection procedures.

B-LEAFS was designed to guide requesters on how they should collect necessary data for training new models. Differently from active learning, the algorithm decides on which feature labels to collect. Integrating these decisions with classical worker / task selection active learning decisions is an open avenue for future work, which would provide end-to-end guidelines in the process of learning in crowdsourcing applications.

5

Troubleshooting Machine Learning Systems via Crowdsourcing

5.1 Overview

Advances in machine learning have enabled the design of integrative systems that perform sophisticated tasks via the execution of analytical pipelines of components. Despite the widespread adoption of such systems, current applications lack the ability to understand, diagnose, and fix their own mistakes which consequently reduces users' trust and limits future improvements. Therefore, the problem of understanding and troubleshooting failures of machine learning systems is of particular interest in the community [135]. Our work studies *component-based* machine learning systems composed of specialized components that are individually trained for solving specific problems and work altogether for solving a single complex task. We analyze how the special characteristics of these integrated learning systems, including *continuous* (non-binary) success measures, *entangled* component design, and *non-monotonic* error propagation, make it challenging to assign blame to individual components. These challenges hinder future system improvements as designers lack an understanding of how different potential fixes on components may improve the overall system output.

The aforementioned challenges are different from the challenges encountered in previous

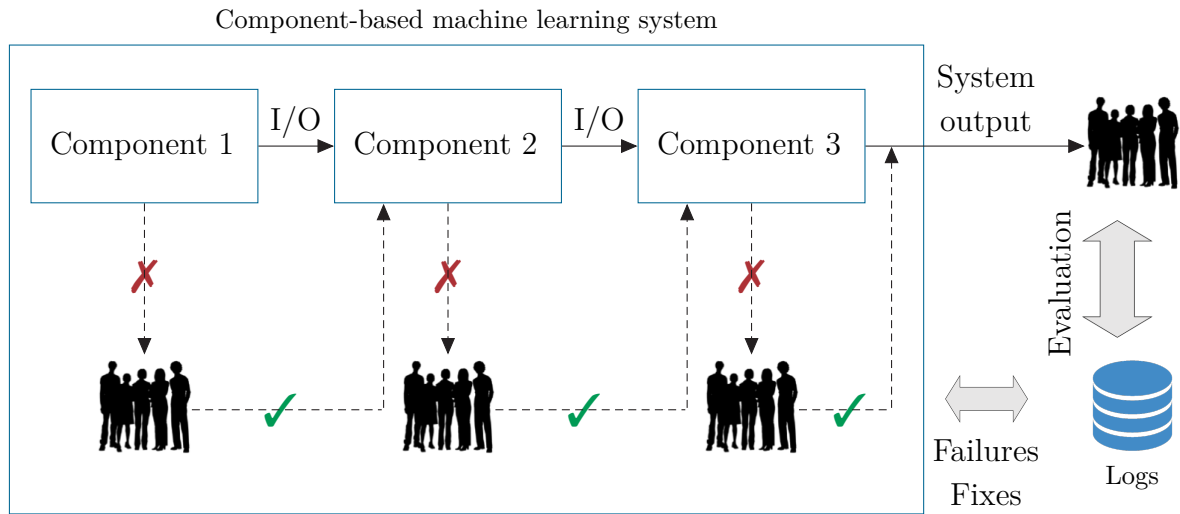


Figure 5.1: *Troubleshooting with humans in the loop*

work for the analogous problem of troubleshooting physical devices. Previous studies on this related problem [144, 24] are based on the following assumptions: (i) the system / component state is binary (normal or faulty) (ii) exactly one of the components is faulty, and (iii) repairing one of the components does not deteriorate the final system state. Instead, we observe that none of these assumptions is true for systems whose components are based on machine learning techniques. Hence, the ambitious goal of our framework is to assist system designers in tackling these challenges by providing them with introspective insights on the system behavior so that these insights can guide them on future system improvements. To the best of our knowledge, this is the first work that studies the troubleshooting problem in the context of component-based machine learning systems by leveraging human input.

Approach. We introduce a troubleshooting methodology which relies on crowdworkers to identify and fix mistakes in existing systems. Human intervention is crucial to the approach as human fixes simulate improved component output that cannot be produced otherwise without significant system development efforts. Figure 5.1 shows the main flow of our approach. First, workers evaluate the system output without any fix to analyze the current system state. To simulate a component fix, the input and output of the component accompanied with the fix description are sent to a crowdsourcing platform as microtasks. Once workers apply the targeted fixes for a component, the fixed output is integrated back into the running system, which thereby generates an improved version of the component. The system is executed as a simulation with the

injected fix and the output is evaluated again via crowdworkers. The overall process collects a valuable set of log data on system failures, human fixes, and their impact on the final output. This data can then be analyzed to identify the most effective combination of component improvements to guide future development efforts.

Case study. We apply our methodology to a state-of-the-art integrative learning system developed to automatically caption images [47]. The system involves three machine learning components in a pipeline, including *visual detection*, *language generation*, and *caption ranking*. The multimodal nature of this case study allows us to demonstrate the applicability of the approach for components processing different forms of data and carrying out various tasks. The methodology reveals new insights on the error dynamics previously unknown to the designers, and offers recommendations on how to best improve the system. For example, in contrast to what system designers had assumed, improving the Reranker is more effective than improving the Visual Detector. Experiments highlight the benefits of making informed decisions about component fixes as their effectiveness varies greatly (18%, 3% and 27% for the three components respectively).

5.2 Related work

Interactive and interpretable machine learning. Previous to integrating crowdsourcing in machine learning processes, there have been prior efforts in interactively including the feedback of machine learning experts in refining current models [45, 162]. A comprehensive summary of the current advances and challenges in interactive machine learning can be found in [13]. The study emphasizes that given the right user interfaces and interaction models, human contribution to machine learning algorithms expands to a broad range of capabilities that go beyond data labeling. Such capabilities include quality assesment, model criticism, error sensitivity specification etc.

Interactive machine learning relies on people being able to understand and interpret machine learning output [80]. In terms of interpreting predictions of classifiers, [130] propose LIME as an explanatory algorithm for any classifier, and SP-LIME as a method for selecting representatory instances for such explanations. The problem is however more convoluted for pipelines of machine learning components where signals from various components are combined to provide a single output to the user. Our work makes a step forward in this direction by providing a methodology that quantifies the current performance of machine learning systems and decomposes it according to the performance of their components.

Crowdsourcing input for machine learning. The contribution of crowdsourcing to machine learning has been mostly limited to the creation of offline data sets for learning (*e.g.*, [101, 140]). Recently, there has been interest in actively integrating crowd participation to the development of machine learning algorithms. Flock [28] builds hybrid crowd-machine classifiers in which crowd workers generate new informative features by reasoning on how instances of positive and negative examples differ from each other. The feature discovery approach presented in [178] asks workers to compare triplets of examples and explain why two out of the three examples are similar to each other. For this purpose, the work also presents an adaptive algorithm for selecting triplets based on previously discovered features. Alloy [26] focuses on clustering text documents by introducing humans in actively sampling examples to support machine learning approaches to overcome their current shortcomings in understanding the semantic differences between items.

Compared to the literature on incorporating human input in developing machine learning systems, there has been only limited work on understanding and diagnosing errors of such systems. On debugging a single classifier, researchers developed an automated method for detecting errors in training data through an interactive debugging process with a domain expert [25]. In a related line of work, researchers proposed an explanatory debugging approach in which the user engages with machine learned systems through two-way interactive explanations, first for understanding machine decisions and then for correcting them [87]. Gestalt is a development environment enabling pipelines that interleave implementation and data analysis to promote bug fixing and discovery [123]. The Beat the Machine game [15], acquires the crowd input for detecting the rare but important errors of predictive models. Our work contributes to this line of literature by studying the diagnosis of component-based systems, rather than individual predictive components, by leveraging the crowd input.

Crowdsourcing for image captioning. Crowdsourcing is heavily used for collecting data for object recognition and image captioning [101, 47]. In terms of improving the acquisition of human input for object recognition, researchers have developed games to elicit discriminative, fine-grained features from the crowd [39]. Given the large-scale annotation effort needed for object recognition, previous work investigated decision-theoretic optimization techniques for deciding when and how human input can complement machine vision algorithms for generating accurate annotations of images [131]. Zhang et al., developed self-assessment models for predicting the failures of vision systems based on input images [172]. Parikh and Zitnick performed user studies to understand what makes humans superior to machines in vision tasks [120].

In terms of evaluating the performance of a component-based automated system, pre-

vious work explored replacing different components with human input to measure the impact of each component on final performance [121]. This approach provides information on the current system but does not offer guidance on possible improvements. Similarly, another work replaced components of an image captioning system to measure upper-bounds on system performance [168]. Our work differs from these studies by offering a general pipeline that simulates and evaluates the effect of different component fixes on system performance to guide system designers on future improvements.

Troubleshooting and blame assignment. The problems of error diagnosis and blame assignment have been studied for systems whose components are not machine learned and the state of components is binary through rule-based and model-based diagnosis approaches [34]. Breese and Heckerman developed Bayesian networks for predicting the state of each component and for making decisions about the next repair action to take [24]. Other researchers investigated decentralized approaches for diagnosis, where each component has a local diagnostic mechanism and their analysis needs to be incorporated for the diagnosis of the system state [30]. In recent work, Sculley et. al., overviewed the challenges of maintaining and improving real-world machine learning systems highlighting error diagnosis as a critical task in particular for component-based systems [135].

An alternative way of improving machine learning algorithms is active learning [138]. Current techniques are applicable to single components (*e.g.* classifiers) but not to integrative systems with multiple components which is the focus of this work. Therefore, active learning methods can be considered as individual component fixes but unfortunately they are not yet applicable to integrative systems with multiple components. Moreover, specific errors within a component oftentimes cannot be fixed by only introducing new training data as the quality improvements for the underlying model may get saturated. Such situations require experimenting with more fundamental and versatile component improvements which we aim to support in our troubleshooting methodology.

5.3 Case study: An image captioning system

5.3.1 System architecture

Visual Detector. The first component of the system is a Visual Detector which recognizes from a given image a list of words associated with recognition scores. The detected words can belong to any part of speech: nouns, verbs, adjectives etc. How-

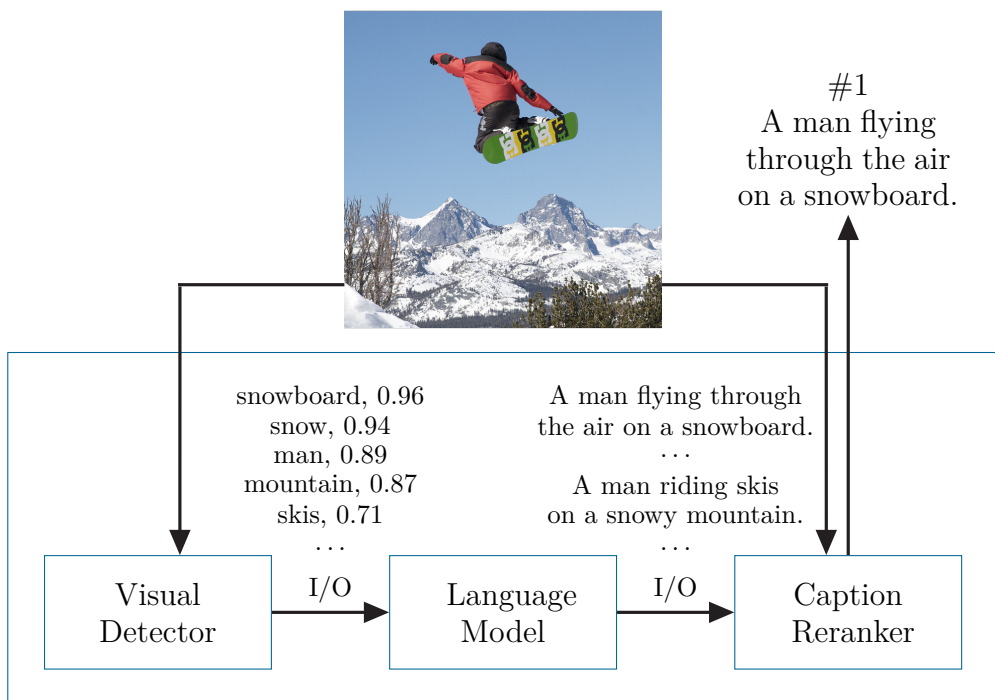


Figure 5.2: The image captioning system

ever, the detector only works on a restricted vocabulary of the 1000 most common words present in the training captions. For this vocabulary, the component learns the presence of each word in the image by using multiple instance learning [171] along with a convolutional neural network applied on overlapping shifted regions of the image. The score of a word then is computed as the maximum score that the word receives across all regions. Note that this method does not differentiate between the various part of speech, which means that the word `elephant` is learned in the same way as the word `running` although the former represents an object while the other one an activity. The list of {word, score} tuples is shortened via a global score threshold and then forwarded to the Language Model.

Language Model. This component is a maximum entropy statistical model [19] that generates likely sentences based on the detected words without having access to the input image. The model uses N-gram features (up to 4-gram) and is trained only on MSCOCO without using any external data. It generates sequences of words, whose loglikelihood score is computed based on the probability of each word conditioned on the other words in the sequence. The statistical nature of this component encodes commonly encountered knowledge in the training data. For instance, the component is able to understand that the word `cat` is more likely to be followed by the word `sleep`

rather than `snowboard`. However, as we show in the experiments, the commonsense awareness of this model is a delicate trade-off between applying statistical knowledge and including words detected from the first component. The set of the 500 most likely word sequences (*i.e.* image captions) and the respective log-likelihood scores are forwarded to the Caption Reranker.

Caption Reranker. The task of the final component is to rerank the captions generated from the Language Model and select the best match for the image. For this purpose, the Reranker is trained only on the M-best lists of sentences. The ranking model uses multiple features among which the log-likelihood of the sentence, the sentence length, the number of objects mentioned in the sentence, and most importantly the deep multimodal similarity of the sentences to the image. This similarity is computed through a Deep Multimodal Similarity Model, which learns two separate neural networks for mapping the images and the text sequences into vector representations and computes their cosine similarity. Both networks are trained with the objective of maximizing the likelihood of a text to describe the image defined in the context of the cosine similarity between the respective vector representations. The caption with the highest likelihood computed via this similarity is finally selected as the output of the system for the given image.

5.3.2 MSCOCO dataset

All components in the system are trained on the publicly available MSCOCO¹ dataset [101] which was also used as part of the large-scale image captioning competition. MSCOCO contains 160,000 images of 91 object types. These objects are generally simple enough to be recognized by a 4 year old person. Moreover, the dataset contains 5 human-generated captions all images in the training and the validation dataset. Each component is individually trained on the image-caption pairs in the training set while we use images from the validation dataset to evaluate our approach.

5.4 Problem characterization

In this section, we define the problem of human-assisted troubleshooting of machine learning systems. First, we describe the problem context along with the required terminology. Next, characterize the troubleshooting problem and the associated challenges that are specific to machine learning systems.

¹<http://mscoco.org/>

5.4.1 Problem context

5.4.1.1 System architecture definition

This work studies machine learning systems consisting of several components designed to carry out specific tasks in the system. The system takes a set of data as *system input* (I) and the individual components work together to produce a final *system output* (O). We assume that the system architecture is provided to the methodology by system designers by specifying:

1. A set of N component nodes $C = \{c_1, \dots, c_N\}$ along with the *component input* and *output* data types.
2. A set of directed communication dependencies between components denoting the input / output exchange. A dependency d_{ij} denotes that the output of c_i serves as an input to c_j . The whole set of dependencies defines the system *execution workflow* (E). In our methodology, we only handle acyclic dependencies but allow for branching. Therefore, the execution workflow can be specified by an ordered set of nodes E whose size is equal to the number of components in the system, $|E| = |C| = N$.

5.4.1.2 Quality definition

For a single system execution, we use an observable and quantifiable quality measure $Q(S, I)$ to represent the quality of the output of system S when executed with I as an input. When a machine learned system solves a complex problem that combines various tasks (*e.g.* image captioning), the quality definition has multiple dimensions and needs to be rethought for every newly introduced system. Often, designers use *automatic scores* as a proxy to compare the system output with a previously known ground truth, which may be generated by humans. These proxy metrics have limitations in assessing system quality as ground truth is not always available and for complex tasks there may be more than one plausible ground truth (*e.g.* an image may have more than one plausible caption). Therefore, in addition to automatic scores, we also evaluate the quality of system output based on a *human satisfaction score* assigned to the caption from crowdsourcing workers, resembling a real-world user evaluation.

For a given component c_i running in a system S with I_i as a component input, we use a quality measure $Q_i(c_i, S, I_i)$ to represent the quality of the component output in the context of the system. Note that this quality measure is dependent on the

context in which the component is being used in S . For example, a visual detector for image captioning has different quality requirements compared to a general-purpose visual detector, as it needs to detect only those objects that are sufficiently prominent to mention in the caption.

5.4.2 Problem definition

Troubleshooting of component-based machine learning systems can be decoupled to answering the following two questions:

Question 1: *How does the system fail?*

The system designer is interested in identifying and measuring the different types of *system failures* and their frequencies as well as the failures of individual components in the system context.

Question 2: *How to improve the system?*

System failures can be addressed by various potential fixes applicable to individual components. To guide future efforts on improving the system, the system designer is interested in knowing the effects of component fixes on the overall system output quality, which provides guidance on future efforts to improve the whole system.

We pose these two questions in the SYSTEM TROUBLESHOOTING problem definition, which is the focus of this chapter.

Problem 5.1 (SYSTEM TROUBLESHOOTING). *Given a component-based machine learning system that consists of N components $C = \{c_1, \dots, c_N\}$, the goal is to analyse system failures and understand the impact of component improvements / fixes in recovering failures and improving the overall system quality.*

5.4.3 Problem characteristics

Next, we examine the special characteristics of component-based machine learning systems that make the problem of troubleshooting challenging. These characteristics differentiate this problem from previous work on troubleshooting and motivate our methodology.

5.4.3.1 Continuous quality measures.

Uncertainty is inherent in machine learning components. When these components work together to solve complex tasks, the quality measure of components and the

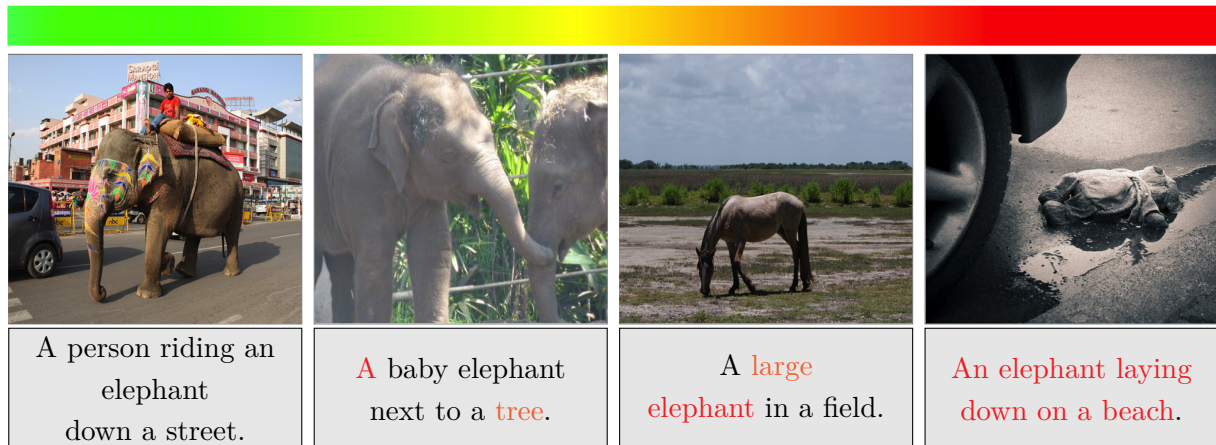


Figure 5.3: Continuous output quality in the image captioning system



Figure 5.4: Continuous output quality in the Visual Detector component

system as a whole is no longer binary, rather it spans a wide spectrum. Therefore, the evaluation of these systems needs to go beyond accuracy metrics to deeper analysis of system behavior. For instance, shows a few examples from image captioning where the system (Figure 5.3) and component output (Figure 5.4) varies in quality and the types of mistakes. Blame assignment in this quality continuum where all components are only partially correct / faulty is non-trivial, which motivates our work for for troubleshooting.

5.4.3.2 Complex component entanglement

In component-based machine learning systems, components have complex influences on each other as they may be tightly coupled or the boundaries between their responsibilities may be not be clear. When the quality of a component depends on the output


|  | Visual Detector | Language Model | Caption Reranker |
|-----------------------------------------------------------------------------------|-------------------------|-------------------------------------------------|-----------------------------------------------------|
| | 1. teddy 0.92 | 1. A teddy bear. | 1. A blender sitting on top of a cake. |
| | 2. on 0.92 | 2. A stuffed bear. | 2. A teddy bear in front of a birthday cake. |
| | 3. cake 0.90 | ... | 3. A cake sitting on top of a blender . |
| | 4. bear 0.87 | | |
| | 5. stuffed 0.85 | | |
| | ... | | |
| | 15. blender 0.57 | 108. A blender sitting on top of a cake. | |

Figure 5.5: *Component entanglement in the image captioning system*

of previous components, blame cannot be assigned to individual components without decoupling imperfection problems in component inputs. Figure 5.5 illustrates a typical scenario of component entanglement in the image captioning system. The final caption is clearly unsatisfactory as it mentions a non-existing object (**blender**). However, the Visual Detector assigns a low score to this word (0.57), which makes the detector only partially responsible for the mistake. The Language Model is also partially responsible as it creates a sentence with low commonsense awareness. Finally, the caption reranker chooses as the best caption a sentence that includes a word with a low score. In this example, the errors from all components are interleaved and it is not possible to disentangle their individual impact on the final error.

5.4.3.3 Non-monotonic error

We note that improving the outputs of components does not guarantee system improvement. On the contrary, doing so may lead to quality deterioration. For example, when components are tuned to suppress erroneous behavior of preceding components, applying fixes to the earlier ones may result to unknown failures. Figure 5.6 shows an example of non-monotonic error behavior. Here, the Visual Detector makes a mistake including **computer** in the list. The initial assumption would be that if the list is fixed so that it contains only the prominent words, then the quality of the caption should increase. In reality, the caption after the fix is more erroneous than the original. Since the language model finds a teddy bear wearing glasses unlikely, it creates a caption that mentions a **person** instead.

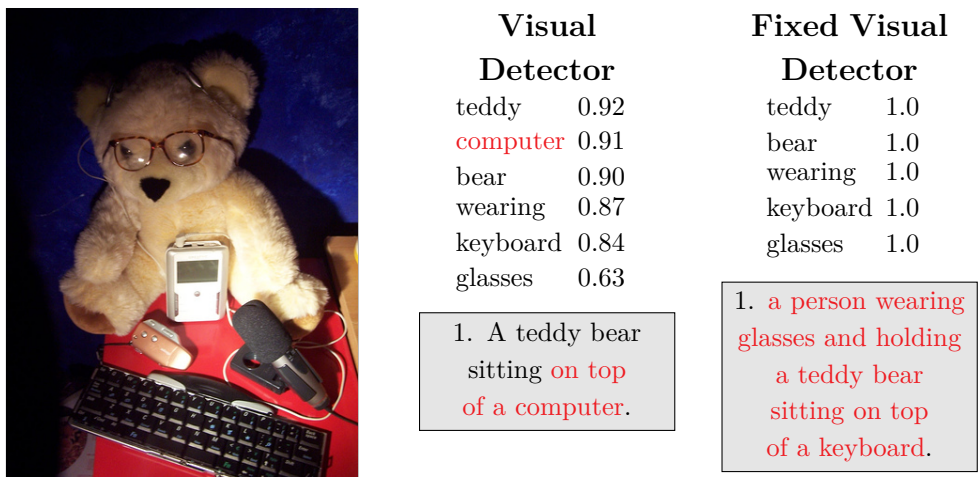


Figure 5.6: Non-monotonic error in the image captioning system

5.5 Human-in-the-loop methodology

Due to these problem characteristics, blame assignment is challenging in integrative systems and analyzing only the current state of the system is not sufficient to develop strategies for system improvement. As shown in Figure 5.1, our methodology overcomes these challenges by introducing humans in the loop for: (i) simulating component fixes, and (2) evaluating the system before and after fixes to directly measure the effect of future system improvements.

5.5.1 Human computation fixes

A component fix is simulated by translating it to a microtask. The microtask exposes the input and the output of the component to crowdsourcing workers and instructs them about the fix within the system scope. In this setting, workers see the component as a black box and do not need to understand the internal details of the component. We denote with $c_i^{(f_k)}$ the process of applying a *human fix* f_k on the output of component c_i . Depending on the component, there may be several human fixes applicable to the same component. In the system level, a *fix workflow* F is a system execution workflow in which the output of the components is corrected via a set of human fixes. For example, the fix workflow $F = \{c_1^{(f_1 f_3)}, c_2, c_3^{(f_4)}\}$ applies the fixes f_1 and f_3 on the first component, no fix in the second component, and f_4 on the third component. After applying a fix workflow, the system and the components' outputs are updated along with the respective quality measures. In addition, a fix workflow execution

consecutively changes the input of the components following those in which a fix was applied.

5.5.2 Methodology setup

The troubleshooting methodology is applicable to systems that follow the assumptions of: (1) *system modularity* with clearly defined component inputs and outputs, and (2) *human interpretability* of the component input / output. To apply the methodology to a new system, the system designer provides the set of components and their input/outputs within the system execution workflow. After identifying a list of component fixes that can potentially improve the system, the system designer formulates corresponding crowdsourcing tasks for these fixes and the overall system evaluation. Both types of tasks should describe the high-level goal of the system, the context in which it operates as well as its requirements (*e.g.* an image captioning system designed to assist users with visual impairments). In addition, component fixing tasks should be appropriately formulated so that their expected output matches the output of implementable fixes that the system designer plans to test.

5.5.3 Troubleshooting steps

The execution of the methodology is guided by the *fix workflow* as a combination of various component fixes to be evaluated. The system designer chooses which fix workflows to execute and evaluate for the purpose of troubleshooting. For a given fix workflow, the steps of our methodology are as follows:

1. *Current system evaluation* — workers assess the final output of the current system on various quality measures.
2. *Component fix simulation* — for each fix in the workflow, workers complete the respective micro-task for examining and correcting the component output.
3. *Fix workflow execution* — executing a fix workflow involves integrating the corrected outputs of each component into the system execution.
4. *After-fix system evaluation* — workers re-evaluate the new system output after incorporating component fixes.

When a workflow includes fixes for multiple components, steps 2 and 3 need to be repeated so that the fixes of earlier components are reflected on the inputs of later components.

5.5.4 Troubleshooting outcomes

Applying human fix workflows simulates improved component states and helps system designers to observe the effect of component fixes on system performance, overcoming the challenges raised by the problem characteristics.

Continuous quality measures Comparing the system quality before and after various fix workflow executions not only can quantify the current quality of system and component output, but it can also isolate and quantify the effect of individual component fixes. For example, if many components are partially failing and possibly responsible for a specific error, the system designer can test the respective fixes, systematically understand their impact, and decide which are the fixes that matter.

Non-monotonic error Non-monotonic error propagation can be disclosed when the overall system quality drops after a component fix. When such a behavior is observed, the system designer can conclude that although such fixes may improve the internal component state, they are not advisable to be implemented in the current state of the system as they produce negative artifacts in the holistic system.

Complex component entanglement Entanglement detection requires the execution of workflows with different combinations of fixes to measure the individual and the joint effect of component fixes. For example, if two consecutive components are entangled, individual fixes in either one of the components may not improve the final output. However, if both components are fixed jointly, this may trigger a significant improvement. The designer could also use this information to detect entanglement and potentially correct the system architecture in future versions.

Most importantly, the complete analysis enables system designers to make informed decisions on choosing the most effective component improvements based on the variation of system improvements after fix workflow executions.

5.6 Troubleshooting the image captioning system

We now describe the customized crowdsourcing tasks for our case study for both system evaluation and component fixes.

5.6.1 System evaluation

The purpose of the system evaluation task is to assign a quality score to the final system output that is close to *human satisfaction*. With this motivation in mind, we suggest a system evaluation where designers include various measures as important quality dimensions in a micro-task. For a consistent comparison, the same evaluation task is then going to be used for evaluating the system both before and after applying human fixes. Our evaluation task for the image captioning system shows workers an image-caption pair in a microtask and ask them to evaluate the following quality measures:

1. *Accuracy* (1-5 Likert scale) — the ability of the caption to mention correct information on the image.
2. *Detail* (1-5 Likert scale) — the ability of the caption to mention important information on the image.
3. *Language* (1-5 Likert scale) — the language fluency of the caption.
4. *Commonsense* (binary) — the ability of the caption to describe images with commonsense awareness.
5. *General evaluation* (1-5 Likert scale) — the general satisfaction of the worker from the image caption.

For each measure, we provided a detailed description along with representative examples. However, we on purpose did not instruct the workers on how to judge image-caption pairs for the general evaluation in order to encourage them to report their real satisfaction from the caption. For instance, we did not bias them on which quality measure is more important (*e.g.* accuracy vs. detail). Also, we clearly stated that a caption may still have a perfect general evaluation even if it fails to have a perfect score on the other measures and vice versa.

5.6.2 Component fixes

Table 5.1 lists all component fixes designed for the system. For all fixes, the task instructions task instructions also describe the system goal to the workers and ask them to apply fixes in the same context. This is an essential detail in the task design because, as we mentioned earlier in the problem definition, the quality measure $Q_i(c, S, I_i)$ for a certain component c depends on the system S where it is employed.

| Component | Fix | Description |
|------------------|--------------|--------------------------------|
| Visual Detector | $c_1^{(f1)}$ | Add objects |
| Visual Detector | $c_1^{(f2)}$ | Remove objects |
| Visual Detector | $c_1^{(f3)}$ | Add activities |
| Visual Detector | $c_1^{(f4)}$ | Remove activities |
| Language Model | $c_2^{(f1)}$ | Remove noncommonsense captions |
| Language Model | $c_2^{(f2)}$ | Remove non-fluent captions |
| Caption Reranker | $c_3^{(f1)}$ | Rerank Top 10 captions |

Table 5.1: Summary of human fixes for the image captioning system.

5.6.2.1 Visual Detector fixes

The visual detector recognizes various part of speech words: nouns, verbs, prepositions etc., and outputs a list of words. Among these part of speech types, mistakes about nouns and verbs can be detected and corrected when workers review the output of the visual recognizer as they can simply tell whether the word should be mentioned in the caption or not. However, this decision is more difficult and in some cases infeasible for relationships and qualifiers as the word should be associated with the respective noun. For instance, in order to understand whether the word **in front of** is applicable to the caption, it needs to mention which object stands in front of which other object. This association is not present in the word list. Therefore, these aspects cannot be properly corrected in visual detector fixes, but they can only be fixed in the caption reranker.

We designed two different tasks for the visual detector, respectively one for fixing objects and another one for fixing activities. In the *object fix* task we show workers the input image together with the list of nouns present in the visual detector output. Workers are asked to correct the list by either removing objects or adding new ones. Since the final list should contain only prominent objects to be mentioned in the caption, workers are instructed to also remove objects that are indeed present in the image but not prominent enough to be mentioned. As the system is able to recognize words from a limited dictionary of 1000 words, sometimes the word that the worker is looking for may not be in the dictionary. In this case, we extend the word search with a word lookup on `wordnet` [49] which maps synonyms and super concepts of the keywords to words that are already in the dictionary. If in the worst case, none of the synonyms or super concepts matches a dictionary word, the workers can add the original keyword. This input can still be useful to the designers for understanding which words are prevalent

5.6. Troubleshooting the image captioning system

enough to be inserted in the dictionary. The *activity task* has a similar design except that now workers are allowed to also submit either an unmodified or an empty list of activities. This is motivated by the fact that the list of detected activities is usually much shorter than for objects. In order to decouple the effect of adding new words to the list from the effect of removing the existing ones, in our analysis we separate both the object and the activity fixes into object / activity addition and removal fixes as shown in Table 5.1. The goal here is to understand how improvements of the Visual Detector precision and recall will influence the final system output. The result of the Visual Detector fixes is a new word list which is forwarded to the Language Model together with the corresponding worker agreement scores (*e.g.* majority vote).

5.6.2.2 Language Model fixes

The goal of this set of fixes is to improve the quality of the output of the language model. We implemented two types of fixes for removing sentences that do not have commonsense awareness and removing those that do not use a fluent English. These fix tasks do not show the input image to the workers and ask workers to make their assessments based on the captions only, as the language model itself does not have access to the input image. In the *commonsense fix* ($c_2^{(f1)}$), workers are required to mark whether a caption describes a likely situation that makes sense in the real world. For example, the caption **a cat playing a video game** has no commonsense awareness. In the *language fix* ($c_2^{(f2)}$) instead, the goal is to mark sentences whose language is not satisfactory in a 1-5 Likert scale. In addition, workers also highlight problematic parts of the sentence which they think would make the caption fluent if fixed appropriately. We use these patterns to prune non-fluent captions that contain the same patterns.

Integrating language model fixes in the system execution removes the noncommonsense and the non-fluent captions from the caption list forwarded to the Caption Reranker.

5.6.2.3 Caption Reranker fixes.

In the Caption Reranker fix, we show workers an image together with the top 10 captions ranked highest in the original Caption Reranker. The presentation order of the captions is randomized so that workers do not get biased from the original ranking. Workers can then pick up to 3 captions that they think fit the image best. However, they can also report that none of the 10 sentences is satisfactory for the image. The workers' feedback is then aggregated via majority vote and the most frequent caption is selected as the best.

5.7 Experimental evaluation

The evaluation of the captioning system with our methodology uses an `Evaluation` dataset of 1000 images randomly selected from the MSCOCO validation dataset. All experiments were performed on Amazon Mechanical Turk.

5.7.1 Quality scores

We report the system quality based on human assessments as well as automatic machine translation scores. The human satisfaction scores are based on the crowdsourced system evaluation. The automatic scores are adapted from automatic machine translation scores where the automatically translated text is compared to human-generated translations. Similarly, in image captioning, an automatic caption is compared to the five image captions retrieved from crowdsourcing workers available in the MSCOCO dataset. While this evaluation is generally cheaper than directly asking people to report their satisfaction, it does not always correlate well with human satisfaction. More specifically, studies show that often what people like is not necessarily similar to what people generate [153]. This phenomena happens due to the fact that an image description can be expressed in many different ways from different people. As a result, designing good automatic scores for image captioning (and machine learning tasks in general) is an active research area [167, 153, 17, 117], as it facilitates systematic and large-scale evaluation of systems.

In our evaluation, we experimented with CIDEr, Bleu4, Bleu1, ROUGEL, and METEOR. The Bleu scores [117] compute n-gram precision similarities of the candidate caption with respect to the reference captions. For example, Bleu4 scores higher those captions that have patterns of 4 consecutive words in common with the human captions. However, Bleu scores do not explicitly model recall which in this case would measure how well does the automatic caption cover the content of the human captions. ROUGE [97] instead, is a recall-based score that was proposed for content summarization which makes it suitable for image captioning as the caption is in fact a textual summary of the visual content in the image. METEOR [17] and CIDEr [153] take into account multiple versions of captions independently and have shown to correlate better with human satisfaction than the other methods.

| Human Satisfaction Scores | | | |
|---------------------------|-------|-------|--------|
| | Eval. | Sat. | Unsat. |
| Accuracy (1-5) | 3.674 | 4.474 | 2.579 |
| Detail (1-5) | 3.563 | 4.265 | 2.601 |
| Language (1-5) | 4.509 | 4.693 | 4.256 |
| Commonsense (0-1) | 0.957 | 1.000 | 0.898 |
| General (1-5) | 3.517 | 4.306 | 2.437 |
| %Satisfactory (0-1) | 57.8% | 100% | 0% |

Table 5.2: *Current system evaluation — Human satisfaction scores.*

| Automatic Scores | | | | |
|------------------|-------|-------|-------|--------|
| | Val. | Eval. | Sat. | Unsat. |
| CIDEr | 0.909 | 0.974 | 1.248 | 0.628 |
| Bleu 4 | 0.293 | 0.294 | 0.368 | 0.184 |
| Bleu 1 | 0.697 | 0.693 | 0.757 | 0.606 |
| ROUGE L | 0.519 | 0.521 | 0.578 | 0.443 |
| METEOR | 0.247 | 0.248 | 0.284 | 0.2 |

Table 5.3: *Current system evaluation — Automatic scores.*

5.7.2 How does the system fail?

5.7.2.1 The current system state

First, we evaluate the current system state as shown in Table 5.2. To gain a deeper understanding of the system performance, we divide the **Evaluation** dataset in two datasets: **Satisfactory** and **Unsatisfactory** based on the general evaluation score collected from the system evaluation task. We consider every answer in 1-3 as an unsatisfactory evaluation, and every other answer in 4-5 as satisfactory. All instances whose original caption reaches a majority agreement on being satisfactory belong to the **Satisfactory** dataset. The rest is classified as **Unsatisfactory**.

Result: Only 57.8% of the images in the **Evaluation** dataset have a satisfactory caption. The comparison between the **Satisfactory** and **Unsatisfactory** partitions shows that the highest discrepancies happen for the *accuracy* and *detail* measures, highlighting the correlation of accuracy and detail with the overall satisfaction.

Since the MSCOCO dataset contains five human generated captions for all the images in the **Validation** dataset (40,504 images), we were able to compute the automatic scores

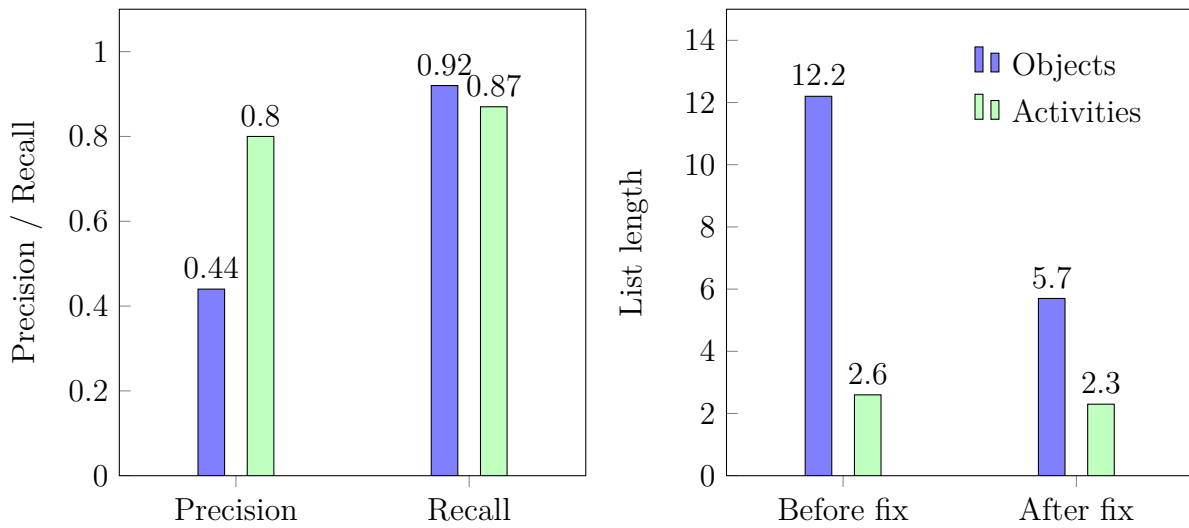


Figure 5.7: *Impact of Visual Detector fixes on the component state*

| | Commonsense fix | Language fix | All fixes |
|-------------|-----------------|--------------|--------------|
| Top1-Eval. | 8.0% | 22.9% | 25.0% |
| Top10-Eval. | 8.6% | 21.7% | 27.1% |
| Top1-Val. | 3.0% | 15.2% | 16.1% |
| Top10-Val. | 2.6% | 14.2% | 14.9% |

Table 5.4: *Percentage of pruned captions from the Language Model.*

for the whole `Validation` dataset and compare it with the scores from `Evaluation` as in Table 5.3.

Result: The evaluation on automatic scores shows that the `Evaluation` dataset is a good representative of the whole `Validation` dataset as it has similar automatic scores. The highest differences between the `Satisfactory` and `Unsatisfactory` datasets are observed for the `CIDEr` and `Bleu4` scores, highlighting their agreement with human satisfaction.

5.7.2.2 The current component state

Next, we also evaluated the current state of each individual system component.

Visual Detector. Figure 5.7 shows the precision and recall of the Visual Detector for both objects and activities when compared to the human-curated lists created from

the majority vote aggregation of multiple workers' fixes. In the same figure, we also show the size of the lists before and after applying human fixes.

Result: The Visual Detector produces longer lists for objects than for activities but with lower precision and recall.

Language Model. In the Language Model fixes, we examine only those captions from the Language Model that are among the Top10 best captions in the Caption Reranker. Given that many of the 500 generated sentences never appear as best captions of the Reranker, and the Language Model output is quite extensive to be fully fixed via crowdsourcing, we focus only on those captions that are likely to impact the system output. Table 5.4 shows the percentage of captions pruned after applying the two fixes. The **Validation** dataset here represents the whole MSCOCO dataset which contains 40,504 images.

Result: Due to self-repetition within the dataset, fixes generated for the 1000 images of the **Evaluation** dataset generalize well to the whole **Validation** set, pruning 16.1% of the Top1 captions and 14.9% of the Top10 captions. Language fixes have a higher coverage than the commonsense ones.

Caption Reranker. Caption Reranker fixes also focus only on the Top10 best captions. After reranking this set with the crowdsourcing majority vote, we observe that the best caption changes for 76.9% of the images. In 46.1% of the cases, the original caption was never chosen by any of the workers. For 19.2% of the images, the majority of workers reported that they could not find any caption in the list that is a good fit for the image. These cases are indeed more serious failures that cannot be recovered through reranking only.

5.7.3 Component fixes

5.7.3.1 Visual Detector fixes

Tables 5.5 and 5.6 show results from applying the four types of fixes on the Visual Detector. These fixes increase the number of satisfactory captions in the dataset up to 17.6% compared to the initial state of the system. Object fixes are more effective than the activity ones for two reasons. First, the precision of the Visual Detector is originally significantly lower for objects than for activities (0.44 vs. 0.8), which offers more room for improvement for object fixes. Second, activity fixes are limited by the shortcomings of the Language Model. Even when a corrected list of activities is provided to the Language Model, it may fail to form commonsense captions containing

| Human Satisfaction Scores — Evaluation dataset | | | | | | |
|------------------------------------------------|--------------|--------------|----------|--------------|---------|--------------|
| | No fix | Object | Activity | Addition | Removal | All fixes |
| Accuracy | 3.674 | 4.045 | 3.681 | 3.709 | 4.000 | 4.035 |
| Detail | 3.563 | 3.900 | 3.590 | 3.604 | 3.880 | 3.916 |
| Language | 4.509 | 4.505 | 4.427 | 4.521 | 4.423 | 4.432 |
| Csense. | 0.957 | 0.947 | 0.940 | 0.957 | 0.933 | 0.942 |
| General | 3.517 | 3.848 | 3.510 | 3.549 | 3.796 | 3.831 |
| %Sat. | 57.8% | 69.1% | 57.1% | 58.5% | 66.8% | 68.0% |

Table 5.5: *Visual Detector fixes — Human satisfaction scores.*

| Automatic Scores — Evaluation dataset | | | | | | |
|---------------------------------------|--------|--------------|----------|----------|---------|--------------|
| | No fix | Object | Activity | Addition | Removal | All fixes |
| CIDEr | 0.974 | 1.048 | 0.948 | 0.995 | 1.023 | 1.045 |
| Bleu4 | 0.294 | 0.298 | 0.289 | 0.299 | 0.289 | 0.295 |
| Bleu1 | 0.693 | 0.713 | 0.690 | 0.698 | 0.711 | 0.719 |
| ROUGEL | 0.521 | 0.529 | 0.517 | 0.524 | 0.524 | 0.528 |
| METEOR | 0.248 | 0.254 | 0.247 | 0.251 | 0.253 | 0.257 |

Table 5.6: *Visual Detector fixes — Automatic scores.*

the corrected activities (*e.g.* A woman holding an office) due to non-monotonic error behavior of the system.

Result: The entangled design between the Visual Detector and the Language Model causes non-monotonic error propagation in particular for activity fixes.

Among all automatic scores, CIDEr, Bleu4, and METEOR preserve the same trends as the human satisfaction score. For example, they confirm that object and removal fixes are more effective than respectively the activity and addition fixes. The non-monotonic error propagations are also reflected in the automatic score analysis as previously concluded from the human satisfaction evaluation.

Improvements for the ROUGEL, METEOR, and Bleu4 are lower (also for other types of fixes) as the metrics require a more complete coverage of the human captions which is challenging to achieve in a sentence with limited length. For example, ROUGEL and METEOR require high recall, while Bleu4 relies on exact matches of 4-gram patterns.

| Human Satisfaction Scores — Evaluation dataset | | | | |
|------------------------------------------------|--------|-------------|----------|--------------|
| | No fix | Commonsense | Language | All fixes |
| Accuracy | 3.674 | 3.698 | 3.696 | 3.712 |
| Detail | 3.563 | 3.583 | 3.590 | 3.602 |
| Language | 4.509 | 4.575 | 4.618 | 4.632 |
| Csense. | 0.957 | 0.973 | 0.974 | 0.982 |
| General | 3.517 | 3.546 | 3.557 | 3.572 |
| %Sat. | 57.8% | 58.5% | 59.2% | 59.3% |

Table 5.7: *Language Model fixes — Human satisfaction scores.*

| Automatic Scores — Evaluation dataset | | | | |
|---------------------------------------|--------|--------------|--------------|--------------|
| | No fix | Commonsense | Language | All fixes |
| CIDEr | 0.974 | 0.975 | 0.983 | 0.98 |
| Bleu4 | 0.294 | 0.297 | 0.297 | 0.298 |
| Bleu1 | 0.693 | 0.694 | 0.690 | 0.691 |
| ROUGEL | 0.521 | 0.524 | 0.526 | 0.527 |
| METEOR | 0.248 | 0.249 | 0.247 | 0.247 |

Table 5.8: *Language Model fixes — Automatic scores.*

5.7.3.2 Language Model

As shown in Tables 5.7 and 5.8, language fixes are generally more effective than the commonsense fixes as they have a higher coverage and they generalize better to other images. Fixes in the Language Model increase the number of satisfactory captions by only 3%.

Result: The impact of Language Model fixes is limited due to the fact that most captions with language mistakes also have other problems which cannot be fixed only through this component.

5.7.3.3 Caption Reranker fixes

As a final component, fixes in the Caption Reranker (Tables 5.9 and 5.10) directly affect the final caption. This means that if there is a plausible caption in the Top10 set better than the original best caption, that caption is going to be ranked higher after the fix and will directly improve the system output. This explains why Caption Reranker improvements are higher than all other component fixes. However, notice

| Human Satisfaction Scores — Evaluation dataset | | |
|------------------------------------------------|--------|-----------------------|
| | No fix | Reranking (All fixes) |
| Accuracy | 3.674 | 4.145 |
| Detail | 3.563 | 3.966 |
| Language | 4.509 | 4.626 |
| Csense. | 0.957 | 0.988 |
| General | 3.517 | 3.973 |
| %Sat. | 57.8% | 73.6% |

Table 5.9: *Caption Reranker fixes — Human satisfaction scores.*

| Automatic Scores — Evaluation dataset | | |
|---------------------------------------|--------|-----------------------|
| | No fix | Reranking (All fixes) |
| CIDEr | 0.974 | 1.087 |
| Bleu4 | 0.294 | 0.320 |
| Bleu1 | 0.693 | 0.720 |
| ROUGEL | 0.521 | 0.543 |
| METEOR | 0.248 | 0.261 |

Table 5.10: *Caption Reranker fixes — Automatic scores.*

that the Caption Reranker fixes are also limited. Fixing the Reranker has no effect if none of the captions in Top10 are satisfactory. This was the case in 19.2% of our dataset. In these cases, fixing the Reranker alone is not sufficient to improve the final caption, the fix workflow needs to include fixes of earlier components as well to improve the final caption.

Result: The system improves by a factor of 27% after the Reranker fixes. Although this provides the most effective system improvement, its influence is limited to instances with at least one satisfactory caption in Top10, which is the case only for 80.8% of our dataset.

5.7.3.4 Complete fix workflow

Tables 5.11 and Tables 5.12 show the improvements from each component and the complete fix workflow which sequentially applies all possible component fixes (*i.e.* $F = \{c_1^{(f_1f_2f_3f_4)}, c_2^{(f_1f_2)}, c_3^{(f_1)}\}$). Figure 5.8 decouples the results for the **Satisfactory** and **Unsatisfactory** partitions of the data set. In Section 5.7.6, we present concrete examples on the impact of of fixes on the system output.

5.7. Experimental evaluation

| Human Satisfaction Scores — Evaluation dataset | | | | | |
|------------------------------------------------|--------|--------------------|-------------------|---------------------|--------------|
| | No fix | Visual Detector | Language Model | Caption Reranker | All fixes |
| Accuracy | 3.674 | 4.035 | 3.712 | 4.145 | 4.451 |
| Detail | 3.563 | 3.916 | 3.602 | 3.966 | 4.247 |
| Language | 4.509 | 4.432 | 4.632 | 4.626 | 4.660 |
| Csense. | 0.957 | 0.942 | 0.982 | 0.988 | 0.998 |
| General | 3.517 | 3.831 | 3.572 | 3.973 | 4.264 |
| %Sat. | 57.8% | 68.0% | 59.3% | 73.6% | 86.9% |

Table 5.11: Complete fix workflow — Human satisfaction Scores.

| Automatic Scores — Evaluation dataset | | | | | |
|---------------------------------------|--------|--------------------|-------------------|---------------------|--------------|
| | No fix | Visual Detector | Language Model | Caption Reranker | All fixes |
| CIDEr | 0.974 | 1.045 | 0.98 | 1.087 | 1.106 |
| Bleu4 | 0.294 | 0.295 | 0.298 | 0.320 | 0.315 |
| Bleu1 | 0.693 | 0.719 | 0.691 | 0.720 | 0.743 |
| ROUGEL | 0.521 | 0.528 | 0.527 | 0.543 | 0.543 |
| METEOR | 0.248 | 0.257 | 0.247 | 0.261 | 0.266 |

Table 5.12: Complete fix workflow — Automatic scores.

Result: The complete fix workflow increases the number of satisfactory captions by 50%. In contrast to the initial assumptions of system designers, fixes in the Caption Reranker are most effective due to the entanglement in the previous components. Most improvements come from the **Unsatisfactory** dataset partition. However, because of non-monotonic error behavior in specific instances, some fixes result in slight deteriorations on the **Satisfactory** partition (*e.g.* Visual Detector fixes).

Moreover, the results show that the complete fix workflow does not improve each quality metric equally. The limited dictionary of the system limits how much the detailedness of captions can improve. Oftentimes, the system fails to make the caption closely specific to the image as the dictionary does not include specific words that are prominent in the image (*e.g.* **wheelchair**) and the correct word is mapped to a word that is already in the dictionary but too general (*e.g.* **pancakes** to **food**).

The comparison of automated scores with human evaluations shows that automated scores are able to detect major improvements in performance from fixes but they fail to recognize the impact of fixes on the various quality dimensions (*i.e.* accuracy, detail

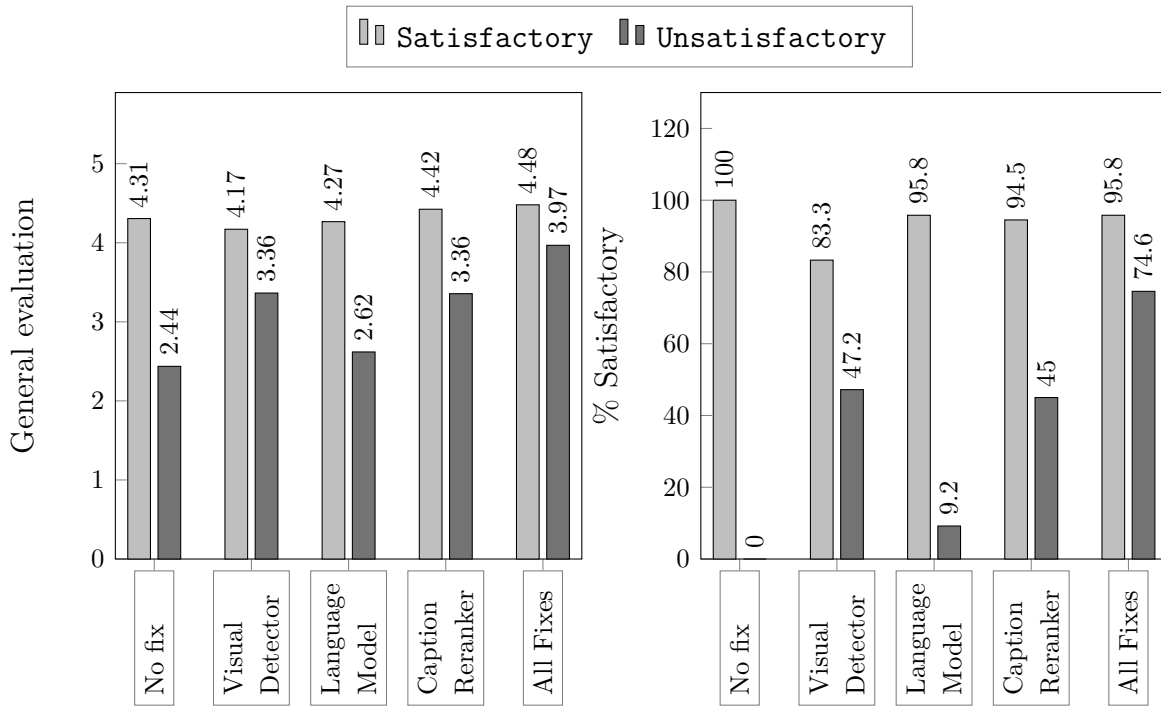


Figure 5.8: Human satisfaction scores on the *Satisfactory* and *Unsatisfactory* dataset partitions.

etc.), showcasing their limitations in comparison to human evaluation. In overall, Caption Reranker fixes remain the most effective ones, followed by the Visual Detector, and the Language Model.

5.7.4 Quality control

For all crowdsourcing experiments we applied the following quality control techniques:

Worker training — Providing accurate instructions to crowdsourcing workers is a crucial aspect in applying our methodology to new machine learning systems. We paid special attention to the task setup by reiterating over the user interface design, providing multiple examples of correct solutions, and giving online feedback to workers on their work quality. Detailed task instructions are necessary for workers to understand the goal of the system and the role of their fixes. We also noticed that workers get more engaged if they understand how their work contributes to improving the presented application.

Spam detection — Due to the multimodal nature of the case study, the set of crowdsourcing tasks we used in our methodology is rich and versatile both in terms of content (*e.g.* images, words, sentences) and design (*e.g.* yes / no questions, word list correction etc.). Therefore, each task required specific spam detection techniques. Given the lack of ground truth, we used worker disagreement as the main criterion to distinguish low-quality work. However, due to potential subjectivity in our tasks, we used worker disagreement only as an initial indication of low quality and further analyzed distinct cases to decide upon work acceptance.

Batching — Large crowdsourcing batches are exposed to low-quality work risk as workers may get tired or reinforce repetitive and wrong biases over time. To avoid such effects, we performed experiments in small-sized batches (*e.g.* 250 image-caption pairs) which were published in periodical intervals. This allowed us to analyze the batch data offline and give constructive feedback to specific workers on how they can improve their work in the future. The strategy also helped to keep workers engaged and motivated.

5.7.5 Methodology cost

The cost of human-in-the loop troubleshooting for a new machine learning system depends on the number of components, the number of fixes, the fix workload, and the size of the dataset to be investigated. Our analysis covered various fix workflows on all components in the 1000 images **Evaluation** dataset which showed to be a good representative of the **Validation** dataset. The total cost of the complete fix workflow (the most expensive one) was \$1,850, respectively spent in system evaluation (\$250), Visual Detector fixes (\$450), Language Model fixes (\$900), and Caption Reranker fixes (\$250). For a more specialized troubleshooting, the system designer can guide the process towards components that are prime candidates for improvement or on errors to which users are most sensitive.

Table 5.13 shows the cost of each crowdsourcing task used in our methodology for the captioning system. Based on this table, system designers can estimate the cost of any future fix workflow. The last column corresponds to the total cost for all instances in the **Evaluation** dataset. Note that the data collected for the Language Model fixes is reusable as the sentences pruned in one workflow can be safely pruned in other workflows as well. Once we executed the tasks for the workflow that fixes only the Language Model, we observed that due to self-repetition, the number of new sentences

| Crowdsourcing task | Fix | Tasks | Cost | Assignments | Total cost |
|------------------------------|------------------|--------------|--------|-------------|------------|
| Visual Detector | | | | | |
| Add / remove objects | $c_1^{(f_{12})}$ | 1000 | \$0.05 | 3 | \$250.00 |
| Add / remove activities | $c_1^{(f_{34})}$ | 1000 | \$0.04 | 3 | \$200.00 |
| Language Model | | | | | |
| Mark noncommonsense captions | $c_2^{(f_1)}$ | ≤ 10000 | \$0.02 | 3 | \$600.00 |
| Remove non-fluent captions | $c_2^{(f_2)}$ | ≤ 10000 | \$0.01 | 3 | \$300.00 |
| Caption Reranker | | | | | |
| Rerank Top 10 captions | $c_3^{(f_1)}$ | 1000 | \$0.05 | 5 | \$250.00 |
| Human satisfaction | | 1000 | \$0.05 | 5 | \$250.00 |
| Maximum workflow cost | | | | | \$1,850.00 |

Table 5.13: Summary of the crowdsourcing cost in the *Evaluation* dataset.

generated from other workflows is at least 4 times lower. This means that the cost of fixes for the Language Model continuously decreases over time.

5.7.6 Examples of fix integration

Figure 5.9 presents examples of different ways fix workflows affect the system output. Figure 5.9(a) is an example of fixes to the Visual Detector resulting in a satisfactory system output. In this example, workers removed the erroneous object `kite` and added `umbrella` which propagated the improvement to the final caption. In the larger dataset, successful propagations of individual component fixes to the final output are also observed for activity fixes, commonsense fixes, and caption refinement fixes. Figure 5.9(b) shows an example of fixes having a limited improvement on the final caption due to the commonsense barrier of the Language Model. In this example, the word `horse` was present in both the original and the fixed word list. However, none of the sentences generated by the Language Model could depict the situation in the image as it was not found to be likely. This example is not unique, the *Unsatisfactory* dataset contains a few more images of the same nature which describe an unlikely situation that are (at the moment) hard to be described by a statistical automated system. Figure 5.9(c) is an example in which improvements from fixes are hindered by the limited size of the dictionary. Since the word `guitar` is not in the dictionary, the final caption fails to provide a satisfactory description.

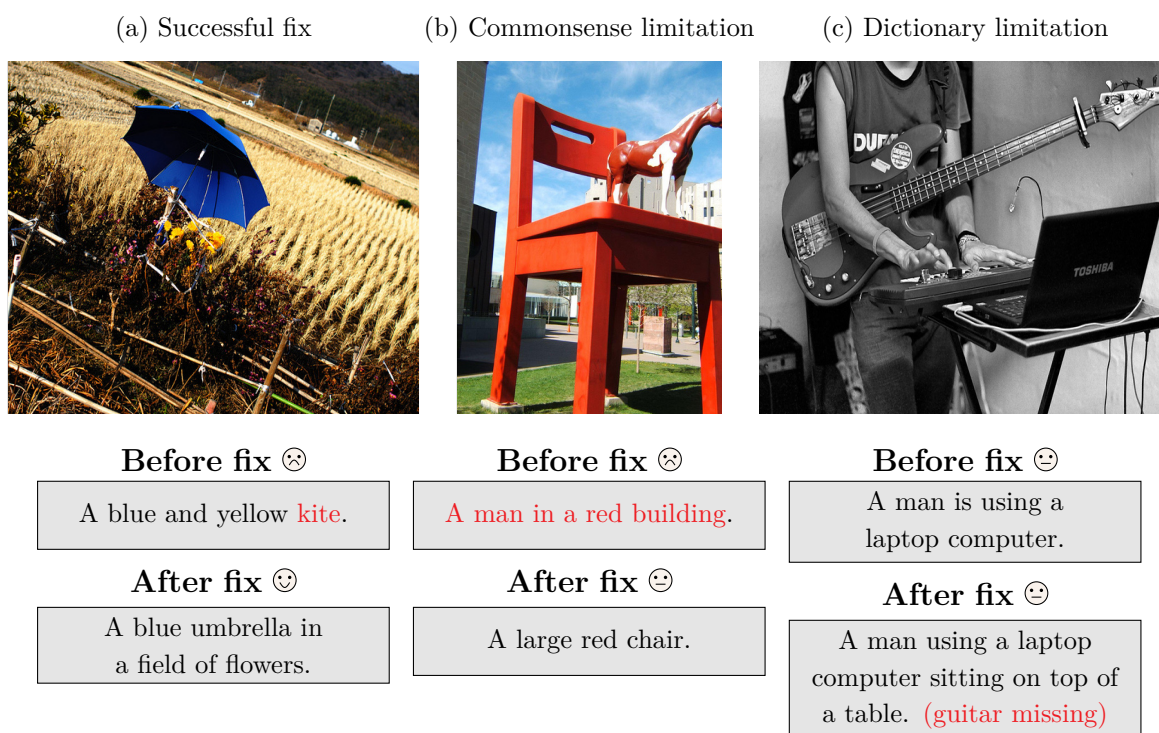


Figure 5.9: Examples of applying the complete fix workflow

5.8 How to improve the system?

Which fixes are more efficient? The results from the methodology provide guidance on next steps to improve the captioning system. First, improving the Reranker emerges as the most promising direction to pursue. Second, due to entanglement issues, improvements on the Visual Detector are suppressed by the shortcomings of the Language Model. Therefore, Visual Detector fixes need to be accompanied with a more capable and commonsense Language Model.

How to implement the fixes? There are multiple ways how human input collected from simulating component fixes can help with permanently implementing component fixes. Human fixes on the Visual Detector reveal that the majority of mistakes are false detections. This issue can be addressed by improving model precision, which can be achieved in multiple ways: (i) adjusting the thresholds used for creating the word list, or (ii) expanding the model to reason about the prominence (saliency) of the word for describing a given image. While the first idea is easier to implement, the latter is not straightforward and the changes would affect a significant part of the detection model.

Moreover, the data collected from language and commonsense fixes can be used for training better language models or for immediately filtering out phrases and sentences flagged by workers. Finally, since a common type of reranking error occurs when the component chooses sentences with words scored low by the Detector, the Reranker can be improved by increasing the weight of the image-caption similarity score.

Can all mistakes be fixed? The human fixes we experimented with in our case study were able to simulate a system state where 86.9% of the captions were satisfactory (Table 5.11). Still, the combination of all fixes we implemented failed on 13.1% of our dataset. We can gain important insights into the shortcomings of the system that cannot be easily fixed by asking the following question: *What are the characteristics of these unrecoverable cases?* Based on an analysis of this subset, we categorize these unrecoverable instances into three groups: (1) images with complex relationships between objects that require more advanced models (Figure 5.9(b)), (2) images representing rare and out-of-domain entities, activities, and relationships that require a broader domain knowledge and a customized training set (Figure 5.9(c)), and (3) images that can be described in many different ways and their captions fail the evaluation test due to human subjectivity. These observations hint that the coverage of human fixes can be extended and that the improvement of such systems is an incremental ongoing process.

5.9 Discussion: Use cases and generalizability

The general methodology we presented can be applied to a broad range of component-based systems that are designed to be modular and their component input / output is human-interpretable. Even in systems in which these assumptions do not hold in the functional component design, a new structure can be discovered by logically separating components in boundaries where data dependencies are guaranteed and the exchanged data can be analyzed by humans.

Applying the methodology to a new system requires the designer to customize the methodology by identifying component fixes, defining system quality measures and designing human computation tasks for evaluation and for simulating component fixes. In addition to the captioning system, we conceptually applied our methodology to two other systems: (i) question answering with knowledge bases and web search [170], and (ii) an email-based reminder for a personal assistant. Our feasibility analysis showed that both applications are compatible with the methodology and highlighted that the most crucial aspect of customization is providing careful and non-ambiguous training

to crowdsourcing workers tailored to the system context. Given the novelty of the proposed human interventions, the resulting crowdsourcing tasks are expected to be different from the typical labeling tasks frequently encountered in today's platforms. Such problems are usually more interesting and engaging to crowdsourcing workers.

5.10 Summary

This chapter summarized our work on troubleshooting component-based machine learning systems. We particularly focused on the system troubleshooting problem. The problem raises two questions: “*How does the system fail?*” and “*How to improve the system?*”. We observe that, for component-based machine learning systems, this problem exhibits particular intricacies not present in physical or conventional software systems: continuous quality measures, non-monotonic error, and complex component entanglement. Due to these characteristics, blame assignment in case of failure (*i.e.* detecting the responsible component for the failure) is challenging and often not feasible. Consequently, we proposed a troubleshooting methodology that uses the simulation of component fixes as a building block for troubleshooting. The execution of fix simulations can generate future hypothetical states of the system, in which one / some of the components are improved. Afterwards, the system designer can compare and evaluate the outcome with the initial state of the system. The analysis is beneficial not only for troubleshooting, but most importantly for making decisions on how to improve the system and where to focus the development effort of machine learning experts.

Human computation is fundamental to our approach as it facilitates the simulation of component fixes. When a component fix cannot be implemented yet or it is too expensive, crowdsourcing workers can check the component output and fix it according to the respective input. These human computation fixes are collected via a crowdsourcing platform in the form of microtasks. This methodology highlights the benefits of deeper integration of crowd input on troubleshooting and improving integrative systems.

We applied our methodology on a real-world system for automatic image captioning. The system combines interesting aspects of visual perception and language understanding, which has recently attracted interest in the machine learning community. The study showed that the human-in-the-loop methodology can reveal insights, problems, and future improvement suggestions previously unknown to the system designers and oftentimes surprising. While our experimental discussion focused mainly on this system, the generic methodology is applicable to any system with modular design and human-interpretable component input / output. Finally, there is an opportunity to

Chapter 5. Troubleshooting Machine Learning Systems via Crowdsourcing

develop generalizable pipelines for automating human-in-the-loop troubleshooting of machine learning systems with reusable crowdsourcing task templates. Such a pipeline would provide valuable insights on system development and create a feedback loop in support of continuous improvement.

6

Conclusions and future work

6.1 Summary and conclusions

Crowdsourcing has empowered the role of human intelligence in the AI loop either by providing label data for training or by correcting possible mistakes in the computation flow. In this dissertation, we referred to these processes as *human supervision* and *human intervention* in machine learning. In particular, we studied *quality control* and *cost optimization* approaches for such human-in-the-loop extensions of machine learning models and systems. Quality control techniques are motivated by two fundamental challenges. First, crowdwork may be subject to human error and ambiguity, which poses limitations in making correct predictions from trained models. Second, current machine learning systems may generate erroneous output of low quality, which is often unpredictable and hard to diagnose. Given the large scale of datasets, cost optimization techniques are then necessary for planning data acquisition under budget constraints.

6.1.1 Human supervision of machine learning models

Training machine learning algorithms from human-generated crowdsourced data has traditionally heavily relied on profiling individual workers for correctly interpreting their answers. In Chapter 2, we emphasized the importance of leveraging crowd models that rely on higher level crowd representations which also account for group-based correlations among workers. We proposed the Access Path model as an alternative of modeling worker correlations and showed that this design comes with various additional

benefits. First, the APM provides accurate predictions with meaningful confidence, which makes it reliable for decision-making. In addition, it can handle noisy and sparse data, which makes it a practical tool for label aggregation in real-world scenarios.

In Chapters 3 and 4, we focused on cost optimization aspects of human supervision. This part of the thesis highlighted the importance of solving the problem in two stages: (i) *testing time* — while collecting data for making new predictions from a given model, and (ii) *training time* — while collecting data for building a new learning model. In Chapter 3, we showed that the design of the Access Path model facilitates cost optimization for new predictions by enabling the adoption of greedy approximation schemes for crowd access optimization. For this purpose, our analysis proves that information gain is a submodular objective function for the model, which can then guarantee theoretical bounds on the approximate solution. Experimental results also indicate that best crowd access plans are highly diverse and they combine answers from multiple access paths.

Optimization techniques for training time data collection face the challenge of model uncertainty due to insufficient available data in early stages. In Chapter 4, we discuss the B-LEAFS algorithm for making decisions on how to acquire labels for building feature-based classification models. For given budget constraints, B-LEAFS is able to identify a set of informative features for the model, and at the same time learn their corresponding parameters. The algorithm operates in a Bayesian framework, and maintains posterior distributions over all model parameters, thereby enabling us to capture the uncertainty in the model parameters about individual features. It makes greedy decisions for selecting the next feature label to acquire by exploiting the submodularity of information gain from a feature, conditioned on the current state of learning. In addition, it effectively balances exploration and exploitation by employing Thompson sampling. We apply the B-LEAFS ideas on both the Naïve Bayes and Access Path model, and in an end-to-end evaluation we finally present the advantages of employing the algorithm on a model that handles noisy crowdsourced labels.

6.1.2 Human intervention in machine learning systems

In the quest of continuously improving machine learning systems, we proposed a human-assisted troubleshooting methodology. The presented methodology is applicable to component-based integrative systems that interleave functionalities and data from various machine learning components in order to provide a single output to the end user. Due to the inherent uncertainty present in machine learning output and the high complexity of such systems, errors in the final output are hard to understand and

diagnose. Our approach overcomes these challenges by introducing human feedback for evaluating and fixing the output of composing components. Via a set of crowdsourcing tasks, workers are presented with pairs of component input and output, which they can correct given the natural capabilities of human intellect to perform such tasks. This feedback simulates new improved component states that are not feasible to generate without significant development effort and time. The simulations and the system evaluation results before and after human component fixes construct a set of valuable log data which can be used by system designers to make strategic decisions on how to further improve the system.

Chapter 5 describes the workings of our approach and how we applied it on a state-of-the-art intelligent system for automatic image captioning. Due to the multimodal nature of the integrated components, crowdsourced fixing tasks are versatile both in terms of design and content. The overall evaluation demonstrated the benefits of the methodology in two aspects. First, the application of individual and joint fixes in a workflow could reveal surprising design facts about the system, previously not known to the designers, including non-monotonic error behavior and complex component entanglement. Second, results on the impact of individual component fixes in the system quality as a whole provided informed guidelines on which components should be prioritized for improvement. Most importantly, the process of implementing the troubleshooting methodology in a real working system, validated the practical opportunities of incorporating crowd input that goes beyond data labeling and blends within learning pipelines as a built-in quality control module.

6.2 Future work

6.2.1 Crowdsourcing platform support and integration

Quality control and in particular label aggregation has been the focus of crowdsourcing research in the last decade. The problem is important not only to the machine learning community but also to other research communities (*e.g.* psychology, sociology, human-computer interaction etc.) that conduct experiments in crowdsourcing platforms. As of now, there is only minimal guidance and support from the platform side on how to interpret the collected data, target the right workers, or optimize the cost of data collection. This implies that almost all crowd requesters need to come up with particular schemes on their own in order to tackle the problems above. While some of the tasks indeed require specific aggregation methods, many others share numerous commonal-

ities, which hints for possible opportunities to integrate generalizable modules within current frameworks that can be leveraged at users' choice. Such modules can be built to provide any of the following functionalities.

Label aggregation Many of the machine learning packages and libraries offer generic implementations of algorithms and models, which crowd requesters need to adopt for handling crowdsourcing input. However, in the crowdsourcing community, various works have proposed valuable aggregation methods, often superior to the standard generic implementations [102, 141, 174, 81, 71, 53]. The Access Path model we proposed in this dissertation is one representative in this family. In order to support the integration of these methods, crowdsourcing frameworks could provide implementable interfaces that allow requesters to either introduce a model of their own or reuse models proposed and implemented by other requesters. The process would be beneficial not only for assisting crowd requesters but also for having a consolidated understanding of the advantages and limitations of current techniques when they are put into practical use.

Worker quality evaluation Quality scores provided by current frameworks mostly focus on the percentage of work being approved by requesters. However, the approval rate is informative only for excluding adversarial low quality work and it does not reveal any insights on workers' level of expertise. The issue can become even more problematic for novice requesters, which have not yet created a reputation in the framework and also have no previous knowledge about the performance of individual workers.

If worker quality evaluation is combined with label aggregation models, similarly to what happens in probabilistical models, it is possible to generate more reliable and informative quality metrics that can express various dimensions of work: accuracy, creativity, responsiveness, time efficiency etc. Certain metrics can then assist requesters to target the right set of workers. In addition, it is interesting to explore how shared models of quality across various requesters can improve targeted crowdsourcing. Most importantly, shared models can also be beneficial in overcoming data sparsity challenges and discovering worker correlations.

Cost control and optimization Due to the lack of systematic techniques for label aggregation and worker quality evaluation, cost control in current platforms is usually static, *i.e.* it does not adapt neither to the type of task nor to the state of the already collected data. The simplest and the most frequent form of cost control in use is the number of assignments per task, which is constant within

a given batch. However, as we showed in this thesis, there is a significant benefit from optimizing data collection according to users' budget constraints either for planning the cost of new predictions or for bounding the cost of generating new models. To implement adaptive data collection algorithms, requesters usually create temporary empty batches of work which they update as soon as tasks are completed from workers. Again, although this technique is commonly employed in research-oriented experiments, there is only little support for other requesters. Therefore, it would be helpful to offer data collection algorithms as batch execution modes, which users can choose to run.

As discussed in Chapters 3 and 4, an important aspect of cost optimization is targeting the right users for the task. Providing meaningful worker quality evaluation metrics as discussed above, is the first requirement towards targeted crowdsourcing. However, due to the challenges that we presented in these chapters (*i.e.* data sparsity and worker correlation) and the way how open crowd marketplaces work (*i.e.* *pull-based* rather than *push-based*), it is not possible to target individual workers in the crowd. For example, workers may not be available or they may be overloaded with work. A viable idea would be to apply a hybrid model of task distribution between pull-based and push-based, where requesters choose to push the task only to a subgroup of workers (*e.g.* based on the access path design in our case) and then workers pull tasks from the batch. In this setting, requesters could still have a coarse-grained control on which workers complete the tasks, and at the same time overcome the inherent problems relevant to accessing individual workers.

6.2.2 Crowdsourcing in future intelligent systems

This thesis was motivated by emerging opportunities of leveraging crowdsourcing to provide human feedback to current learning systems. We showed that human feedback can be introduced in analytical models either for the purpose of training or making interventions for correcting erroneous output. As the complexity of the machine learning systems and the problems that they solve grows, integrating a human perspective in the design of intelligent pipelines can help towards continuously improving such systems, understanding their complexity, and adapting them to their users' needs. Next, we review some of the open directions, which can be promising in bridging the gap between human and machine computation.

Training for complex problems The problems that current machine learning algorithms can solve are mainly characterized by individual tasks whose solution can be represented via a limited set of labels or actions, which can be directly used as training data. Simple examples include text classification, image recognition etc. Recently, there has been increasing interest in solving more sophisticated problems that involve multiple learning algorithms. The image captioning scenario that we discussed in Chapter 5 is a typical example. Other applications involve machine translation, question answering, and dialogue systems. Current solutions in these domains use crowdsourced data as input, which is usually presented in the form of possible solutions to the task (*e.g.* human translated text, answers, and conversations). This type of training data goes beyond single-label answers. Often, it may even require training by providing a sequence of actions that lead to the correct solution. In this context, human input can be valuable if integrated with other forms of learning, *e.g.* reinforcement learning [108, 147] with implicit or explicit human feedback. The shift has major implications in several aspects of crowdsourcing such as task-design, aggregation models as well as worker profiling.

System evaluation for complex problems An additional input for improving a running system is to provide feedback on how satisfactory the system behavior is to the end user. Most of the system evaluation surveys focus on estimating single quality scores of user satisfaction. However, similar surveys can be conducted to have a detailed understanding of other quality dimensions like: (i) the sensitivity of users to various types of errors, and (ii) their preferences on which task instances should have a higher priority than others. This type of evaluation can guide designers in adjusting current systems according to users' sensitivity. Possible adjustments include the correction of penalty / reward measures of the models so that they can account for online human criticism.

Control transfer In mission-critical applications of machine learning, failure detection and prevention are central prerequisites for building robust and reliable systems [132]. In this thesis, we showed how crowdsourcing can be used as part of troubleshooting methodologies when failures happen. However, it is also equally important for such systems to be able to understand in an online fashion when such failures happen and transfer the further operation flow to users. Designing smooth control transfers is however a challenging problem as it requires the implementation of error-awareness modules that can initiate the process. On the other hand, systems should be sufficiently transparent to users to enable them

understand the failure and take over the control flow. Operation transparency and interpretability are dependent on shared human-machine cognitive models, which can ensure that users build a correct perception of the system and vice versa.

Appendices



Access path selection

A.1 Proof of Theorem 1

In order to prove Theorem 1, we will consider a generic Bayesian Network for the Access Path Model (APM) with N access paths and each access path associated with M possible votes from workers. Hence, we have following set of random variables to represent this network:

- i) Y is the random variable of the crowdsourcing task.
- ii) $Z : \{Z_1, \dots, Z_i, \dots, Z_N\}$ are the latent random variables of the N access paths.
- iii) $X : \{X_{ij} \text{ for } i \in [1, \dots, N] \text{ and } j \in [1, \dots, M]\}$ represents a set of random variables associated with all the workers from the access paths.

The goal is to prove the submodularity property of the set function:

$$f(S) = IG(S; Y) \tag{A.1}$$

i.e., the information gain of Y and $S \subseteq X$ w.r.t to set selection S , earlier referred to as *access plan*. We begin by proving the following Lemma 1 that establishes the submodularity of the information gain in a network with one access path (*i.e.*, $N = 1$), denoted as Z_1 .

Lemma 1. *The set function $f(S) = IG(S; Y)$ in Equation A.1 is submodular for the Bayesian Network representing an Access Path Model with $N = 1$ access path denoted by Z_1 , associated with M workers denoted by $X : \{X_{1j} \text{ for } j \in [1, \dots, M]\}$.*

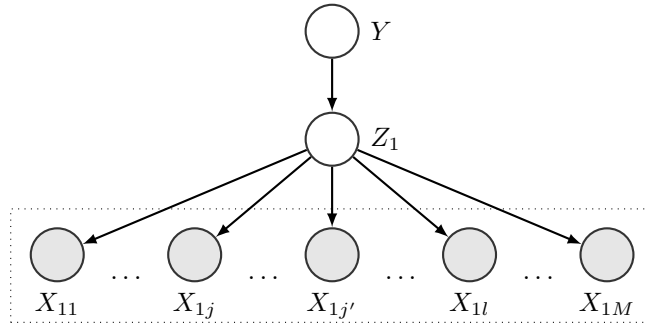


Figure A.1: APM Model for $N = 1$ access path, associated with M workers

Proof of Lemma 1. Figure A.1 illustrates the Bayesian Network considered here with one access path Z_1 . For the sake of the proof, we consider an alternate view of the same network as shown in Figure A.2. Here, the auxiliary variable Z_{1j} denotes the set of first j variables associated with workers' votes from access path Z_1 , *i.e.*, $Z_{1j} = \{X_{11}, X_{12}, \dots, X_{1j}\}$. This alternate view is taken from the following generative process: Z_1 is first sampled given Y , followed by sampling of Z_{1M} from Z_1 , where $Z_{1M} = \{X_{11}, X_{12}, \dots, X_{1M}\}$. Given Z_{1M} , the remaining $Z_{1j} \forall j \leq M$ are just subsets of Z_{1M} . We define set $Q : \{Z_{1j} \text{ for } j \in [1, \dots, M]\}$.

One crucial property we use while considering this generative process here is that all the X_{1j} are just repeated observations of same variable associated with response of a worker from Z_1 access path and hence they are anonymous and ordering does not matter. Note that, querying j workers from Z_1 , *i.e.* observing $S = \{X_{11} \dots X_{1j}\}$ is equivalent to observing Z_{1j} . Given this equivalence of the two representations of Figure A.1 and Figure A.2, we now prove the submodularity of the set function $g(A) = IG(A; Y)$ *i.e.*, the information gain of Y and $A \subseteq Q$ w.r.t to set selection A .

Note that since $Z_{1j} \subseteq Z_{1j'} \forall j \leq j'$, we can alternatively write down A as equivalent to the singleton set given by $\{Z_{1k}\}$ where $k = \arg \max_j Z_{1j} \in A$. Also note that, function $f(S)$ and $g(A)$ have one to one equivalence given by $g(A) = f(\{X_{11} \dots X_{1k}\})$ where $k = \arg \max_j Z_{1j} \in A$.

To prove submodularity of g , consider sets $A \subset A' \subset Q$ and an element $q \in Q \setminus A'$. Let $A \equiv \{Z_{1j}\}$, $A' \equiv \{Z_{1j'}\}$ where $j' > j$ and $q = Z_{1l}$ where $l > j'$. First, let us consider

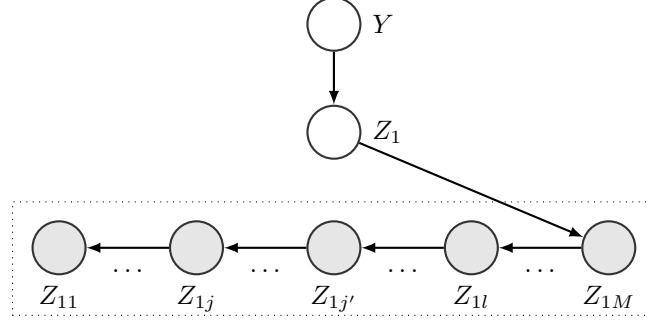


Figure A.2: APM Model for $N = 1$ access path, associated with M workers represented with auxiliary variables Z_{ij}

marginal utility of q over A denoted as $\Delta_g(q|A)$, given by:

$$\begin{aligned}
\Delta_g(q|A) &= g(A \cup \{q\}) - g(A) \\
&= IG(A \cup \{q\}; Y) - IG(A; Y) \\
&= IG(\{Z_{1j}\} \cup \{Z_{1l}\}; Y) - IG(\{Z_{1j}\}; Y) \\
&= IG(\{Z_{1l}\}; Y) - IG(\{Z_{1j}\}; Y) \tag{A.2}
\end{aligned}$$

$$\begin{aligned}
&= IG(Z_{1l}; Y) - IG(Z_{1j}; Y) \tag{A.3} \\
&= \left(H(Y) - H(Y|Z_{1l}) \right) - \left(H(Y) - H(Y|Z_{1j}) \right) \\
&= H(Y|Z_{1j}) - H(Y|Z_{1l})
\end{aligned}$$

Step A.2 uses the fact that $\{Z_{1j}\} \cup \{Z_{1l}\}$ is simply equivalent to $\{Z_{1l}\}$ as $Z_{1j} \subset Z_{1l}$. Step A.3 replaces singleton sets $\{Z_{1l}\}$ and $\{Z_{1j}\}$ by the associated random variables Z_{1l} and Z_{1j} . Now, to prove submodularity, we need to show that $\Delta_g(q|A) \geq \Delta_g(q|A')$, given by:

$$\begin{aligned}
&\Delta_g(q|A) - \Delta_g(q|A') \\
&= \left(H(Y|Z_{1j}) - H(Y|Z_{1l}) \right) - \left(H(Y|Z_{1j'}) - H(Y|Z_{1l}) \right) \\
&= H(Y|Z_{1j}) - H(Y|Z_{1j'}) \\
&= \left(H(Y) - H(Y|Z_{1j'}) \right) - \left(H(Y) - H(Y|Z_{1j}) \right) \\
&= IG(Z_{1j'}; Y) - IG(Z_{1j}; Y) \\
&\geq 0 \tag{A.4}
\end{aligned}$$

Step A.4 uses the “data processing inequality” [18, 31], which states that post-processing cannot increase information, or the mutual information gain between two random vari-

Appendix A. Access path selection

ables decreases with addition of more intermediate random variables in the unidirectional network considered in Figure A.2. \square

Next, we use the result of Lemma 1 to prove the results for generic networks with N access paths.

Proof of Theorem 1. We now consider a generic Bayesian Network for the Access Path Model (APM) with N access paths and each access path associated with M possible votes from workers. Again taking the alternate view as illustrated in Figure A.2, we define auxiliary variables Z_{ij} denoting a set of first j variables associated with workers' votes from access path Z_i , *i.e.*, $Z_{ij} = \{X_{i1}, X_{i2}, \dots, X_{ij}\}$. As before, we define set $Q : \{Z_{ij} \text{ for } i \in [1, \dots, N] \text{ and } j \in [1, \dots, M]\}$. The goal is to prove the submodularity over the set function $g(A) = IG(A; Y)$ *i.e.*, the information gain of Y and $A \subseteq Q$ w.r.t to set selection A .

We define $Q_i : \{Z_{ij} \text{ for } j \in [1, \dots, M]\} \forall i \in [1, \dots, N]$, and hence we can write $Q = \cup_{i=1}^N Q_i$. We can similarly write $A = \cup_{i=1}^N A_i$ where $A_i = A \cap Q_i$. We denote complements of A_i and Q_i as A_i^c and Q_i^c respectively, defined as follows: $Q_i^c = Q \setminus Q_i$ and $A_i^c = A \cap Q_i^c$.

To prove the submodularity property of g , consider two sets $A \subset Q$, and $A' = A \cup \{s\}$, as well as an element $q \in Q \setminus A'$. Let $q \in Q_i$. We consider following two cases:

Case i). $s \in Q_i$ (q and s belong to the same access path.)

Note that, we can write $A = A_i \cup A_i^c$ and $A' = A_i' \cup A_i^c$, as A and A' differ only along access path i . Also, let us denote a particular realization of the variables in set A_i^c by a_i^c . The key idea that we use is that for a given realization of A_i^c , the generic Bayesian Network with N access paths can be factorized in a similar way as with just one access path (Figure A.2), when computing the marginal gains of q over A_i and $A_i \cup \{s\}$.

Again, we need to show $\Delta_g(q|A) \geq \Delta_g(q|A')$; given by:

$$\begin{aligned} \Delta_g(q|A) - \Delta_g(q|A') &= \Delta_g(q|A_i \cup A_i^c) - \Delta_g(q|A_i' \cup A_i^c) \\ &= \mathbb{E}_{a_i^c} \left(\Delta_g(q|A_i, a_i^c) - \Delta_g(q|A_i', a_i^c) \right) \end{aligned} \tag{A.5}$$

$$\geq 0 \tag{A.6}$$

Step A.5 considers expectation over all the possible realizations of random variables in A_i^c . Step A.6 uses the result of Lemma 1 as this network for a given realization of A_i^c has the same characteristics as a single access path network where information

gain is submodular. Hence, each term inside the expectation is non-negative, proving therefore the desired result.

Next, we consider the other case when q and s belong to different access paths.

(Case ii) $s \in Q_i^c$ (q and s belong to different access paths.) First, let us consider marginal utility of q over A denoted as $\Delta_g(q|A)$, given by:

$$\begin{aligned} \Delta_g(q|A) &= g(A \cup \{q\}) - g(A) \\ &= IG(A \cup \{q\}; Y) - IG(A; Y) \\ &= \left(H(A \cup \{q\}) - H(A \cup \{q\}|Y) \right) - \left(H(A) - H(A|Y) \right) \\ &= \left(H(A \cup \{q\}) - H(A) \right) - \left(H(A \cup \{q\}|Y) - H(A|Y) \right) \\ &= H(q|A) - H(q|A; Y) \end{aligned} \tag{A.7}$$

$$= H(q|A) - H(q|A_i; Y) \tag{A.8}$$

Step A.7 simply replaces the singleton set $\{q\}$ with the random variable q . Step A.8 uses the fact that $A = A_i \cup A_i^c$ and the conditional independence of q and A_i^c given Y .

Now, to prove submodularity, we need to show $\Delta_g(q|A) \geq \Delta_g(q|A')$, given by:

$$\begin{aligned} \Delta_g(q|A) - \Delta_g(q|A') &= \left(H(q|A) - H(q|A_i, Y) \right) - \left(H(q|A') - H(q|A_i, Y) \right) \end{aligned} \tag{A.9}$$

$$\begin{aligned} &= H(q|A) - H(q|A') \\ &\geq 0 \end{aligned} \tag{A.10}$$

Step A.9 uses the conditional independence of q and A_i^c given Y . Note that a crucial property used in this step is that $s \in A_i^c$ for this case. Step A.10 follows from the "information never hurts" principle [31] thus proving the desired result and completing the proof. \square

A.2 Proof of Theorem 2

Proof of Theorem 2. In order to prove Theorem 2, we first consider a general submodular set function and prove the approximation guarantees for the greedy selection scheme under the assumption that the cost to budget ratio is bounded by γ .

Let V be a collection of sets and consider a monotone, non-negative, submodular set function f defined over V as $f : 2^V \rightarrow \mathbb{R}$. Each element $v \in V$ is associated with a

Appendix A. Access path selection

ALGORITHM 8. GREEDY for general submodular function

```

1 Input: budget  $B$ , set  $V$ , function  $f$ 
2 Output: set  $S^{\text{GREEDY}}$ 
3 Initialization: set  $S = \emptyset$ , iterations  $r = 0$ , size  $l = 0$ 
4 while  $V \neq \emptyset$  do
5    $v^* = \arg \max_{v \subseteq V} \left( \frac{f(S \cup v) - f(S)}{c_v} \right)$ 
6   if  $c(S) + c_{v^*} \leq B$  then
7      $S = S \cup \{v^*\}$ 
8      $l = l + 1$ 
9    $V = V \setminus \{v^*\}$ 
10   $r = r + 1$ 
11  $S^{\text{GREEDY}} = S$ 
12 return  $S^{\text{GREEDY}}$ 

```

non-negative cost c_v . The budgeted optimization problem can be cast as:

$$S^* = \arg \max_{S \subseteq V} f(S) \text{ subject to } \sum_{s \in S} c_s \leq B$$

Let S^{OPT} be the optimal solution set for this maximization problem, which is intractable to compute [48]. Consider the generic GREEDY selection algorithm given by Algorithm 8 and let S^{GREEDY} be the set returned by this algorithm. We now analyze the performance of GREEDY and start by closely following the proof structure of [79, 146]. Note that every iteration of the Algorithm 8 can be classified along two dimensions: i) whether a selected element v^* belongs to S^{OPT} or not, and ii) whether v^* gets added to set S or not. First, let us consider the case when v^* belongs to S^{OPT} , however was not added to S because of violation of budget constraint. Let r be the total iterations of the algorithm so far, and l be the size of S at this iteration. We can renumber the elements of V so that v_i is the i^{th} element added to S for $i \in [1, \dots, l]$ and v_{l+1} is the first element from S^{OPT} selected by the algorithm that could not be added to S . Let S_i be the set obtained when first i elements have been added to S . Also, let $c(S)$ denote $\sum_{s \in S} c_s$. By using the result of [79, 146], the following holds:

$$f(S_i) - f(S_{i-1}) \geq \frac{c_i}{B} \cdot \left(f(S^{\text{OPT}}) - f(S_{i-1}) \right)$$

Using the above result, [79, 146] shows the following through induction:

$$\begin{aligned} f(S_l) &\geq \left(1 - \prod_{j=1}^l \left(1 - \frac{c_j}{B}\right)\right) \cdot f(S^{\text{OPT}}) \\ &\geq \left(1 - \left(1 - \sum_{j=1}^l \frac{c_j}{B \cdot l}\right)^l\right) \cdot f(S^{\text{OPT}}) \end{aligned} \quad (\text{A.11})$$

$$= \left(1 - \left(1 - \frac{c(S_l)}{B \cdot l}\right)^l\right) \cdot f(S^{\text{OPT}}) \quad (\text{A.12})$$

In Step A.11, we use the property that every function of form $\left(1 - \prod_{j=1}^l \left(1 - \frac{c_j}{B}\right)\right)$ achieves its minimum at $\left(1 - (1 - \beta)^l\right)$ for $\beta = \sum_{j=1}^l \frac{c_j}{B \cdot l}$.

Now, we will incorporate our assumption of bounded costs, *i.e.*, $c_v \leq \gamma \cdot B \forall v \in V$, where $\gamma \in (0, 1)$ to get the desired results. We use the fact that budget spent by Algorithm 8 at iteration r when it could not add an element to solution is at least $(B - \max_{v \subseteq V} c_v)$, which is lower-bounded by $B(1 - \gamma)$. Hence, the cost of greedy solution set $c(S_l)$ at this iteration is at least $B(1 - \gamma)$. Incorporating this in Step A.12, we get:

$$\begin{aligned} f(S_l) &\geq \left(1 - \left(1 - \frac{(1 - \gamma)}{l}\right)^l\right) \cdot f(S^{\text{OPT}}) \\ &= \left(1 - \left(1 - \frac{1}{\eta}\right)^{\eta \cdot (1 - \gamma)}\right) \cdot f(S^{\text{OPT}}) \text{ where } \eta = \frac{l}{(1 - \gamma)} \\ &\geq \left(1 - \frac{1}{e^{(1 - \gamma)}}\right) \cdot f(S^{\text{OPT}}) \end{aligned} \quad (\text{A.13})$$

This proves that the GREEDY in Algorithm 8 achieves a utility of at least $\left(1 - \frac{1}{e^{(1 - \gamma)}}\right)$ times that obtained by optimal solution OPT. Given these results, Theorem 2 follows directly given the submodularity properties of the considered optimization function. \square

List of Tables

| | | |
|-----|---------------------------------------------------------------------------------------------------------------------|-----|
| 1.1 | Quality control and cost optimization problems | 3 |
| 2.1 | Comparison of the APM with current approaches. | 14 |
| 2.2 | Access path configuration for Example 1 | 15 |
| 2.3 | Symbol description for the Access Path model | 21 |
| 2.4 | Accuracy rates and votes for each worker. | 22 |
| 2.5 | Access Path Design for MedicalQA dataset. | 33 |
| 2.6 | UCI datasets description. | 33 |
| 3.1 | Set of answers collected from two access paths for Example 3.1. | 45 |
| 3.2 | Probability of each strategy to correctly solve each of the questions. | 46 |
| 3.3 | Expected accuracy of each strategy. | 46 |
| 4.1 | Example of running B-LEAFS on the <code>nursery</code> dataset for $B = 200$ and $K = 4$ | 72 |
| 4.2 | Example of running all the algorithms on the <code>breast-cancer</code> dataset for $B = 100$ and $K = 2$ | 78 |
| 5.1 | Summary of human fixes for the image captioning system. | 100 |
| 5.2 | Current system evaluation — Human satisfaction scores. | 103 |
| 5.3 | Current system evaluation — Automatic scores. | 103 |
| 5.4 | Percentage of pruned captions from the Language Model. | 104 |
| 5.5 | Visual Detector fixes — Human satisfaction scores. | 106 |

List of Tables

| | | |
|------|-----------------------------------------------------------------------------|-----|
| 5.6 | Visual Detector fixes — Automatic scores. | 106 |
| 5.7 | Language Model fixes — Human satisfaction scores. | 107 |
| 5.8 | Language Model fixes — Automatic scores. | 107 |
| 5.9 | Caption Reranker fixes — Human satisfaction scores. | 108 |
| 5.10 | Caption Reranker fixes — Automatic scores. | 108 |
| 5.11 | Complete fix workflow — Human satisfaction Scores. | 109 |
| 5.12 | Complete fix workflow — Automatic scores. | 109 |
| 5.13 | Summary of the crowdsourcing cost in the Evaluation dataset. | 112 |

List of Figures

| | | |
|------|---------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1 | Hybrid Human-Machine Learning Systems | 2 |
| 2.1 | The APM for crowdsourcing a medical question | 15 |
| 2.2 | Naïve Bayes Individual - NBI. | 21 |
| 2.3 | Naïve Bayes Model for Access Paths - NBAP. | 25 |
| 2.4 | Bayesian Network Model for Access Paths - APM. | 26 |
| 2.5 | Parameters θ of the Access Path Model. | 27 |
| 2.6 | Accuracy with unconstrained budget. | 34 |
| 2.7 | Accuracy and negative log-likelihood for equally distributed budget across all access paths in the MedicalQA dataset. | 35 |
| 2.8 | Accuracy and negative log-likelihood for equally distributed budget across all access paths in the ProbabilitySports dataset. | 36 |
| 2.9 | Uniform noise impact on prediction error for UCI datasets. | 38 |
| 2.10 | Biased noise impact on prediction error for UCI datasets. | 39 |
| 3.1 | Information gain and budget distribution for ProbabilitySports (year=2002). | 52 |
| 3.2 | Crowd access optimization results for varying budget in CUB-200. | 53 |
| 3.3 | Crowd access optimization results for varying budget in ProbabilitySports (year = 2002). | 54 |
| 3.4 | Diversity impact on optimization. Varying access path correlation in MedicalQA. | 55 |
| 3.5 | Diversity impact on optimization. Varying budget in MedicalQA. | 56 |

List of Figures

| | | |
|------|-------------------------------------------------------------------------------------------------------------------------------------|-----|
| 4.1 | Example of feature-based crowdsourced predictions. | 61 |
| 4.2 | The Naïve Bayes model — plate representation. | 64 |
| 4.3 | The Access Path model — plate representation. | 65 |
| 4.4 | Example of binary parameter distributions | 70 |
| 4.5 | Learning and feature selection on Naïve Bayes. Varying B | 79 |
| 4.6 | Learning and feature selection on Naïve Bayes. Varying K | 79 |
| 4.7 | Learning and feature selection on the Access Path model. Synthetic noisy crowd. | 80 |
| 4.8 | Learning and feature selection on the Access Path model. Real-world crowd. | 81 |
| 4.9 | Uniform noise impact on the prediction error of models build from the data collected from B-LEAFS and the whole dataset. | 82 |
| 4.10 | Biased noise impact on the prediction error of models build from the data collected from B-LEAFS and the whole dataset. | 83 |
| 5.1 | Troubleshooting with humans in the loop | 86 |
| 5.2 | The image captioning system | 90 |
| 5.3 | Continuous output quality in the image captioning system | 94 |
| 5.4 | Continuous output quality in the Visual Detector component | 94 |
| 5.5 | Component entanglement in the image captioning system | 95 |
| 5.6 | Non-monotonic error in the image captioning system | 96 |
| 5.7 | Impact of Visual Detector fixes on the component state | 104 |
| 5.8 | Human satisfaction scores on the Satisfactory and Unsatisfactory dataset partitions. | 110 |
| 5.9 | Examples of applying the complete fix workflow | 113 |
| A.1 | APM Model for $N = 1$ access path, associated with M workers | 128 |
| A.2 | APM Model for $N = 1$ access path, associated with M workers repre- sented with auxiliary variables Z_{ij} | 129 |

Bibliography

- [1] Amazon mechanical turk. <https://www.mturk.com/>. Accessed: 2016-07-10.
- [2] citizen science.gov. <https://www.citizen science.gov//>. Accessed: 2016-07-18.
- [3] Crowdfunder. <https://www.crowdfunder.com/>. Accessed: 2016-07-10.
- [4] Galaxy zoo. <https://www.galaxyzoo.org/>. Accessed: 2016-07-18.
- [5] Upwork. <https://www.upwork.com/>. Accessed: 2016-07-10.
- [6] *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.
- [7] Ittai Abraham, Omar Alonso, Vasilis Kandyas, Rajesh Patel, Steven Shelford, and Aleksandrs Slivkins. How many workers to ask? adaptive exploration for collecting high quality labels. *arXiv preprint arXiv:1411.0149*, 2014.
- [8] Ittai Abraham, Omar Alonso, Vasilis Kandyas, and Aleksandrs Slivkins. Adaptive crowdsourcing algorithms for the bandit survey problem. In *COLT 2013*, pages 882–910, 2013.
- [9] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *COLT*, pages 39.1–39.26, 2012.
- [10] Mohammad Allahbakhsh, Boualem Benatallah, A Ignjatovic, HR Motahari-Nezhad, E Bertino, and S Dustdar. Quality control in crowdsourcing systems. *IEEE Internet Comput*, 17(2):76–81, 2013.
- [11] Omar Alonso. Practical lessons for gathering quality labels at scale. In *SIGIR*, 2015.

Bibliography

- [12] Omar Alonso, Daniel E Rose, and Benjamin Stewart. Crowdsourcing for relevance evaluation. In *ACM SigIR Forum*, volume 42, pages 9–15. ACM, 2008.
- [13] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014.
- [14] Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(1):2773–2832, 2014.
- [15] Josh Attenberg, Panagiotis G Ipeirotis, and Foster J Provost. Beat the machine: Challenging workers to find the unknown unknowns. *Human Computation*, 11:11, 2011.
- [16] Krisztian Balog, Yi Fang, Maarten de Rijke, Pavel Serdyukov, and Luo Si. Expertise retrieval. *Foundations and Trends in Information Retrieval*, 6(2–3):127–256, 2012.
- [17] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL*, volume 29, pages 65–72, 2005.
- [18] Normand J. Beaudry and Renato Renner. An intuitive proof of the data processing inequality. *Quantum Info. Comput.*, 12(5-6):432–441, 2012.
- [19] Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.
- [20] Arpita Biswas, Shweta Jain, Debmalya Mandal, and Y Narahari. A truthful budget feasible multi-armed bandit mechanism for crowdsourcing time critical tasks. In *AAMAS*, pages 1101–1109, 2015.
- [21] Alessandro Bozzon, Marco Brambilla, and Stefano Ceri. Answering search queries with crowdsearcher. In *Proceedings of the 21st international conference on World Wide Web*, pages 1009–1018. ACM, 2012.
- [22] Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Matteo Silvestri, and Giuliano Vesci. Choosing the right crowd: expert finding in social networks. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 637–648. ACM, 2013.

- [23] Jonathan Bragg, Daniel S Weld, et al. Optimal testing for crowd workers. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 966–974. International Foundation for Autonomous Agents and Multiagent Systems, 2016.
- [24] John S Breese and David Heckerman. Decision-theoretic troubleshooting: A framework for repair and experiment. In *UAI*, 1996.
- [25] Aleksandar Chakarov, Aditya Nori, Sriram Rajamani, Shayak Sen, and Deepak Vijaykeerthy. Debugging machine learning tasks. *arXiv preprint arXiv:1603.07292*, 2016.
- [26] Joseph Chee Chang, Aniket Kittur, and Nathan Hahn. Alloy: Clustering with crowds and computation. In *CHI*, 2016.
- [27] Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 193–202. ACM, 2013.
- [28] Justin Cheng and Michael S Bernstein. Flock: Hybrid crowd-machine learning classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 600–611. ACM, 2015.
- [29] Justin Cheng and Michael S. Bernstein. Flock: Hybrid crowd-machine learning classifiers. In *CSCW*, pages 600–611, 2015.
- [30] Luca Console, Claudia Picardi, and Daniele Theseider Dupré. A framework for decentralized qualitative model-based diagnosis. In *IJCAI*, pages 286–291, 2007.
- [31] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.
- [32] Peng Dai, Christopher H Lin, Daniel S Weld, et al. Pomdp-based control of workflows for crowdsourcing. *Artificial Intelligence*, 202:52–85, 2013.
- [33] Peng Dai, Mausam, and Daniel S. Weld. Artificial intelligence for artificial intelligence. In *AAAI*, 2011.
- [34] Adnan Darwiche. Model-based diagnosis under real-world constraints. *AI magazine*, 21(2):57, 2000.

- [35] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.
- [36] Luis M De Campos. A scoring function for learning bayesian networks based on mutual information and conditional independence tests. *JMLR*, 7:2149–2187, 2006.
- [37] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.
- [38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE, 2009.
- [39] Jia Deng, Jonathan Krause, Michael Stark, and Li Fei-Fei. Leveraging the wisdom of the crowd for fine-grained recognition. 2015.
- [40] Kun Deng, Chris Bourke, Stephen D. Scott, Julie Sunderman, and Yaling Zheng. Bandit-based algorithms for budgeted learning. In *ICDM*, pages 463–468, 2007.
- [41] Kun Deng, Yaling Zheng, Chris Bourke, Stephen D. Scott, and Julie Masciale. New algorithms for budgeted learning. *Machine Learning*, 90(1):59–90, 2013.
- [42] Wenkui Ding, Tao Qin, Xu-Dong Zhang, and Tie-Yan Liu. Multi-armed bandit with budget constraint and variable costs. In *AAAI*, 2013.
- [43] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Integrating conflicting data: the role of source dependence. *Proceedings of the VLDB Endowment*, 2(1):550–561, 2009.
- [44] Shayan Doroudi, Ece Kamar, Emma Brunskill, and Eric Horvitz. Toward a learning science for complex crowdsourcing tasks. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2623–2634. ACM, 2016.
- [45] Jerry Alan Fails and Dan R Olsen Jr. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 39–45. ACM, 2003.
- [46] Ju Fan, Meihui Zhang, Stanley Kok, Meiyu Lu, and Beng Chin Ooi. Crowdop: Query optimization for declarative crowdsourcing systems. *IEEE Transactions on Knowledge and Data Engineering*, 27(8):2078–2092, 2015.

-
- [47] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. From captions to visual concepts and back. In *CVPR*, pages 1473–1482, 2015.
- [48] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45:314–318, 1998.
- [49] Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.
- [50] Michael J Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 61–72. ACM, 2011.
- [51] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 721–741, 1984.
- [52] Gagan Goel, Afshin Nikzad, and Adish Singla. Allocating tasks to workers with matching constraints: truthful mechanisms for crowdsourcing markets. In *WWW*, pages 279–280. ACM, 2014.
- [53] Ryan G Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. Crowd-clustering. In *Advances in neural information processing systems*, pages 558–566, 2011.
- [54] Bryce Goodman and Seth Flaxman. Eu regulations on algorithmic decision-making and a "right to explanation". *arXiv preprint arXiv:1606.08813*, 2016.
- [55] Anja Gruenheid, Besmira Nushi, Tim Kraska, Wolfgang Gatterbauer, and Donald Kossmann. Fault-tolerant entity resolution with the crowd. *arXiv preprint arXiv:1512.00537*, 2015.
- [56] Stephen Guo, Aditya Parameswaran, and Hector Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 385–396. ACM, 2012.
- [57] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *JMLR*, 2003.
- [58] Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *ICML*, 2000.

Bibliography

- [59] Chien-Ju Ho, Shahin Jabbari, and Jennifer W Vaughan. Adaptive task assignment for crowdsourced classification. In *Proc. of the 30th ICML*, pages 534–542, 2013.
- [60] Chien-Ju Ho, Aleksandrs Slivkins, Siddharth Suri, and Jennifer Wortman Vaughan. Incentivizing high quality crowdwork. In *Proceedings of the 24th International Conference on World Wide Web*, pages 419–429. ACM, 2015.
- [61] Chien-Ju Ho and Jennifer Wortman Vaughan. Online task assignment in crowdsourcing markets. In *AAAI*, volume 12, pages 45–51, 2012.
- [62] Lu Hong and Scott E Page. Groups of diverse problem solvers can outperform groups of high-ability problem solvers. *Proceedings of the National Academy of Sciences of the United States of America*, 101(46):16385–16389, 2004.
- [63] Damon Horowitz and Sepandar D Kamvar. The anatomy of a large-scale social search engine. In *Proceedings of the 19th international conference on World wide web*, pages 431–440. ACM, 2010.
- [64] John Joseph Horton and Lydia B Chilton. The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM conference on Electronic commerce*, pages 209–218. ACM, 2010.
- [65] Jeff Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.
- [66] Ting Wu Lei Chen Pan Hui and Chen Jason Zhang Weikai Li. Hear the whole story: Towards the diversity of opinion in crowdsourcing markets. *Proceedings of the VLDB Endowment*, 8(5), 2015.
- [67] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67. ACM, 2010.
- [68] Panos Ipeirotis and Evgeniy Gabrilovich. Quizz: Targeted crowdsourcing with a billion (potential) users. In *WWW*, 2014.
- [69] Ece Kamar. Directions in hybrid intelligence: Complementing ai systems with human intelligence.
- [70] Ece Kamar, Severin Hacker, and Eric Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International*

-
- Conference on Autonomous Agents and Multiagent Systems- Volume 1*, pages 467–474. International Foundation for Autonomous Agents and Multiagent Systems, 2012.
- [71] Ece Kamar, Ashish Kapoor, and Eric Horvitz. Identifying and accounting for task-dependent bias in crowdsourcing. In *Third AAAI Conference on Human Computation and Crowdsourcing*, 2015.
- [72] Aloak Kapoor and Russell Greiner. *Learning and classifying under hard budgets*. Springer, 2005.
- [73] David R Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal crowdsourcing using low-rank matrix approximations. In *49th Annual Allerton Conference*, pages 284–291. IEEE, 2011.
- [74] David R Karger, Sewoong Oh, and Devavrat Shah. Efficient crowdsourcing for multi-class labeling. *ACM SIGMETRICS Performance Evaluation Review*, 41(1):81–92, 2013.
- [75] David R Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research*, 62(1):1–24, 2014.
- [76] Gabriella Kazai, Jaap Kamps, and Natasa Milic-Frayling. The face of quality in crowdsourcing relevance labels: Demographics, personality and labeling accuracy. In *Proceedings of the 21st ACM CIKM*, pages 2583–2586. ACM, 2012.
- [77] Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from noisy entries. *Journal of Machine Learning Research*, 11(Jul):2057–2078, 2010.
- [78] Raghunandan H Keshavan, Sewoong Oh, and Andrea Montanari. Matrix completion from a few entries. In *2009 IEEE International Symposium on Information Theory*, pages 324–328. IEEE, 2009.
- [79] Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.
- [80] Been Kim. *Interactive and interpretable machine learning models for human machine collaboration*. PhD thesis, Massachusetts Institute of Technology, 2015.
- [81] Hyun-Chul Kim and Zoubin Ghahramani. Bayesian classifier combination. In *AISTATS*, pages 619–627, 2012.

Bibliography

- [82] Aniket Kittur, Susheel Khamkar, Paul André, and Robert Kraut. Crowdweaver: visually managing complex crowd work. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1033–1036. ACM, 2012.
- [83] Aniket Kittur, Jeffrey V Nickerson, Michael Bernstein, Elizabeth Gerber, Aaron Shaw, John Zimmerman, Matt Lease, and John Horton. The future of crowd work. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 1301–1318. ACM, 2013.
- [84] Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E Kraut. Crowdforge: Crowdsourcing complex work. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 43–52. ACM, 2011.
- [85] Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *UAI*, 2005.
- [86] Andreas Krause and Carlos Guestrin. A note on the budgeted maximization of submodular functions. Technical Report CMU-CALD-05-103, Carnegie Mellon University, 2005.
- [87] Todd Kulesza, Simone Stumpf, Margaret Burnett, Weng-Keen Wong, Yann Riche, Travis Moore, Ian Oberst, Amber Shinsel, and Kevin McIntosh. Explanatory debugging: Supporting end-user debugging of machine-learned programs. In *VL/HCC*, pages 41–48. IEEE, 2010.
- [88] Anand Kulkarni, Matthew Can, and Björn Hartmann. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1003–1012. ACM, 2012.
- [89] Matt J. Kusner, Wenlin Chen, Quan Zhou, Zhixiang Eddie Xu, Kilian Q. Weinberger, and Yixin Chen. Feature-cost sensitive learning with submodular trees of classifiers. In *AAAI*, pages 1939–1945, 2014.
- [90] PJ Lamberson and Scott E Page. Optimal forecasting groups. *Management Science*, 58(4):805–810, 2012.
- [91] Edith Law and Luis von Ahn. Human computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(3):1–121, 2011.
- [92] Matthew Lease. On quality control and machine learning in crowdsourcing. *Human Computation*, 11(11), 2011.

-
- [93] AK Leung and C PION Kao. Evaluation and management of the child with speech delay. *American family physician*, 59(11):3121–8, 1999.
- [94] Hongwei Li, Bo Zhao, and Ariel Fuxman. The wisdom of minority: discovering and targeting the right group of workers for crowdsourcing. In *Proceedings of the 23rd international conference on World wide web*, pages 165–176. ACM, 2014.
- [95] Hongwei Li, Bo Zhao, and Ariel Fuxman. The wisdom of minority: Discovering and targeting the right group of workers for crowdsourcing. In *Proc. of the 23rd WWW*, 2014.
- [96] M. Lichman. UCI machine learning repository, 2013.
- [97] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain, 2004.
- [98] Christopher H Lin and Daniel S Weld. Re-active learning: Active learning with relabeling. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- [99] Christopher H Lin, Daniel S Weld, et al. To re (label), or not to re (label). In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.
- [100] Tian Lin, Jian Li, and Wei Chen. Stochastic online greedy learning with semi-bandit feedbacks. In *NIPS*, pages 352–360, 2015.
- [101] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV 2014*, pages 740–755. Springer, 2014.
- [102] Qiang Liu, Jian Peng, and Alexander T Ihler. Variational inference for crowdsourcing. In *NIPS*, pages 701–709, 2012.
- [103] Daniel J. Lizotte, Omid Madani, and Russell Greiner. Budgeted learning of naive-bayes classifiers. In *UAI*, pages 378–385, 2003.
- [104] Daniel Lowd and Pedro M. Domingos. Naive bayes models for probability estimation. In *ICML*, pages 529–536, 2005.
- [105] Adam Marcus, Eugene Wu, David Karger, Samuel Madden, and Robert Miller. Human-powered sorts and joins. *Proceedings of the VLDB Endowment*, 5(1):13–24, 2011.

Bibliography

- [106] Adam Marcus, Eugene Wu, David R Karger, Samuel Madden, and Robert C Miller. Crowdsourced databases: Query processing with people. CIDR, 2011.
- [107] Geoffrey McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.
- [108] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [109] Todd K Moon. The expectation-maximization algorithm. *Signal processing magazine, IEEE*, 13(6):47–60, 1996.
- [110] Barzan Mozafari, Purna Sarkar, Michael Franklin, Michael Jordan, and Samuel Madden. Scaling up crowd-sourcing to very large datasets: a case for active learning. *Proceedings of the VLDB Endowment*, 8(2):125–136, 2014.
- [111] Bonnie M Muir. Trust in automation: Part i. theoretical issues in the study of trust and human intervention in automated systems. *Ergonomics*, 37(11):1905–1922, 1994.
- [112] G.L. Nemhauser, L.A. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Math. Prog.*, 14:265–294, 1978.
- [113] Besmira Nushi, Adish Singla, Anja Gruenheid, Erfan Zamanian, Andreas Krause, and Donald Kossmann. Crowd access path optimization: Diversity matters. In *Third AAAI Conference on Human Computation and Crowdsourcing*, 2015.
- [114] Besmira Nushi, Adish Singla, Andreas Krause, and Donald Kossmann. Learning and feature selection under budget constraints in crowdsourcing. In *Fourth AAAI Conference on Human Computation and Crowdsourcing*, 2016.
- [115] David Oleson, Alexander Sorokin, Greg P. Laughlin, Vaughn Hester, John Le, and Lukas Biewald. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. In *AAAI*, 2011.
- [116] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419, 2010.

-
- [117] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL, 2002.
- [118] Aditya Parameswaran, Stephen Boyd, Hector Garcia-Molina, Ashish Gupta, Neoklis Polyzotis, and Jennifer Widom. Optimal crowd-powered rating and filtering algorithms. *VLDB*, 2014.
- [119] Devi Parikh and C Zitnick. Human-debugging of machines. *NIPS WCSSWC*, 2:7, 2011.
- [120] Devi Parikh and C Lawrence Zitnick. The role of features, algorithms and data in visual recognition. In *CVPR*, pages 2328–2335. IEEE, 2010.
- [121] Devi Parikh and C Lawrence Zitnick. Finding the weakest link in person detectors. In *CVPR*, pages 1425–1432. IEEE, 2011.
- [122] Hyunjung Park, Hector Garcia-Molina, Richard Pang, Neoklis Polyzotis, Aditya Parameswaran, and Jennifer Widom. Deco: A system for declarative crowdsourcing. *Proceedings of the VLDB Endowment*, 5(12):1990–1993, 2012.
- [123] Kayur Patel, Naomi Bancroft, Steven M Drucker, James Fogarty, Andrew J Ko, and James Landay. Gestalt: integrated support for implementation and analysis in machine learning. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology*, pages 37–46. ACM, 2010.
- [124] Ravali Pochampally, Anish Das Sarma, Xin Luna Dong, Alexandra Meliou, and Divesh Srivastava. Fusing data with correlations. In *SIGMOD*, 2014.
- [125] Matteo Pozzetti, Andreas Krause, Donald Kossmann, and Besmira Nushi. Access path design for quality assurance in crowdsourcing. 2016.
- [126] <http://www.probabilitysports.com>.
- [127] Vikas C Raykar and Shipeng Yu. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *Journal of Machine Learning Research*, 13(Feb):491–518, 2012.
- [128] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(Apr):1297–1322, 2010.

Bibliography

- [129] Theodoros Rekatsinas, Xin Luna Dong, Lise Getoor, and Divesh Srivastava. Finding quality in quantity: The challenge of discovering valuable sources for integration. In *CIDR*, 2015.
- [130] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?": Explaining the predictions of any classifier. *arXiv preprint arXiv:1602.04938*, 2016.
- [131] Olga Russakovsky, Li-Jia Li, and Li Fei-Fei. Best of both worlds: human-machine collaboration for object annotation. In *CVPR*, pages 2121–2131, 2015.
- [132] Stuart Russell, Daniel Dewey, and Max Tegmark. Research priorities for robust and beneficial artificial intelligence. *arXiv preprint arXiv:1602.03506*, 2016.
- [133] Karl-Michael Schneider. Techniques for improving the performance of naive bayes for text classification. In *CICLing*, pages 682–693. Springer, 2005.
- [134] François Schnitzler, Jia Yuan Yu, and Shie Mannor. Sensor selection for crowd-sensing dynamical systems. In *AISTATS*, pages 829–837, 2015.
- [135] D Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *NIPS*, 2015.
- [136] P Griffiths Selinger, Morton M Astrahan, Donald D Chamberlin, Raymond A Lorie, and Thomas G Price. Access path selection in a relational database management system. In *Proc. of the 1979 ACM SIGMOD*, pages 23–34. ACM, 1979.
- [137] B. Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.
- [138] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.
- [139] Uri Shaham, Xiuyuan Cheng, Omer Dror, Ariel Jaffe, Boaz Nadler, Joseph Chang, and Yuval Kluger. A deep learning approach to unsupervised ensemble learning. *JMLR*, 2016.
- [140] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *KDD*, pages 614–622. ACM, 2008.

-
- [141] Edwin Simpson, Stephen Roberts, Ioannis Psorakis, and Arfon Smith. Dynamic bayesian combination of multiple imperfect classifiers. In *Decision Making and Imperfection*, pages 1–35. Springer, 2013.
- [142] Adish Singla and Andreas Krause. Truthful incentives in crowdsourcing tasks using regret minimization mechanisms. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1167–1178. ACM, 2013.
- [143] Adish Singla, Sebastian Tschiatschek, and Andreas Krause. Noisy submodular maximization via adaptive sampling with applications to crowdsourced image collection summarization. In *AAAI*, 2016.
- [144] Eleni Stroulia, Murali Shankar, Ashok Goel, and Louise Penberthy. A model-based approach to blame-assignment in design. In *Artificial Intelligence in Design 92*, pages 519–537. Springer, 1992.
- [145] James Surowiecki. *The wisdom of crowds*. Anchor, 2005.
- [146] Maxim Sviridenko. A note on maximizing a submodular set function subject to knapsack constraint. *Operations Research Letters*, v.(32):41–43, 2004.
- [147] Andrea Lockerd Thomaz and Cynthia Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *AAAI*, volume 6, pages 1000–1005, 2006.
- [148] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.
- [149] Long Tran-Thanh, Trung Dong Huynh, Avi Rosenfeld, Sarvapali D Ramchurn, and Nicholas R Jennings. Crowdsourcing complex workflows under budget constraints. 2015.
- [150] Long Tran-Thanh, Sebastian Stein, Alex Rogers, and Nicholas R Jennings. Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. *Artificial Intelligence*, 214:89–111, 2014.
- [151] Long Tran-Thanh, Matteo Venanzi, Alex Rogers, and Nicholas R Jennings. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *AAMAS*, pages 901–908, 2013.
- [152] Jennifer Wortman Vaughan. Adaptive task assignment for crowdsourced classification. 2013.

Bibliography

- [153] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575, 2015.
- [154] Sriharsha Veeramachaneni, Emanuele Olivetti, and Paolo Avesani. Active sampling for detecting irrelevant features. In *ICML*, pages 961–968, 2006.
- [155] Alfredo Vellido, José David Martín-Guerrero, and Paulo JG Lisboa. Making machine learning models interpretable. In *ESANN*, volume 12, pages 163–172. Citeseer, 2012.
- [156] Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. Community-based bayesian aggregation models for crowdsourcing. In *WWW*, pages 155–164. ACM, 2014.
- [157] Matteo Venanzi, W. T. Luke Teacy, Alex Rogers, and Nicholas R. Jennings. Bayesian modelling of community-based multidimensional trust in participatory sensing under data sparsity. In *IJCAI*, 2015.
- [158] Greg Ver Steeg and Aram Galstyan. Discovering structure in high-dimensional data through correlation explanation. In *Advances in Neural Information Processing Systems*, pages 577–585, 2014.
- [159] Jeroen Vuurens, Arjen P de Vries, and Carsten Eickhoff. How much spam can you take? an analysis of crowdsourcing results to increase accuracy. In *Proc. ACM SIGIR Workshop on Crowdsourcing for Information Retrieval*, pages 21–26, 2011.
- [160] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11):1483–1494, 2012.
- [161] Jing Wang and Panagiotis Ipeirotis. Quality-based pricing for crowdsourced workers, 2013.
- [162] Malcolm Ware, Eibe Frank, Geoffrey Holmes, Mark Hall, and Ian H Witten. Interactive machine learning: letting users build classifiers. *International Journal of Human-Computer Studies*, 55(3):281–292, 2001.
- [163] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

-
- [164] Peter Welinder, Steve Branson, Pietro Perona, and Serge J Belongie. The multidimensional wisdom of crowds. In *Advances in neural information processing systems*, pages 2424–2432, 2010.
- [165] Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier R Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, volume 22, pages 2035–2043, 2009.
- [166] Yan Yan, Glenn M Fung, Rómer Rosales, and Jennifer G Dy. Active learning from crowds. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1161–1168, 2011.
- [167] Yezhou Yang, Ching Lik Teo, Hal Daumé III, and Yiannis Aloimonos. Corpus-guided sentence generation of natural images. In *EMNLP*, pages 444–454. ACL, 2011.
- [168] Li Yao, Nicolas Ballas, Kyunghyun Cho, John R Smith, and Yoshua Bengio. Trainable performance upper bounds for image and video captioning. *arXiv preprint arXiv:1511.04590*, 2015.
- [169] Jinfeng Yi, Rong Jin, Anil K Jain, and Shaili Jain. Crowdclustering with sparse pairwise labels: A matrix completion approach. In *AAAI Workshop on Human Computation*, volume 2. Citeseer, 2012.
- [170] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL*, 2015.
- [171] Cha Zhang, John C Platt, and Paul A Viola. Multiple instance boosting for object detection. In *NIPS*, pages 1417–1424, 2005.
- [172] Peng Zhang, Jiuling Wang, Ali Farhadi, Martial Hebert, and Devi Parikh. Predicting failures of vision systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3566–3573, 2014.
- [173] Xiaohang Zhang, Guoliang Li, and Jianhua Feng. Crowdsourced top-k algorithms: An experimental evaluation. *Proceedings of the VLDB Endowment*, 9(8):612–623, 2016.
- [174] Yuchen Zhang, Xi Chen, Denny Zhou, and Michael I Jordan. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. In *Advances in neural information processing systems*, pages 1260–1268, 2014.

Bibliography

- [175] Yaling Zheng, Stephen Scott, and Kun Deng. Active learning from multiple noisy labelers with varied costs. In *2010 IEEE International Conference on Data Mining*, pages 639–648. IEEE, 2010.
- [176] Denny Zhou, Sumit Basu, Yi Mao, and John C Platt. Learning from the wisdom of crowds by minimax entropy. In *Advances in Neural Information Processing Systems*, pages 2195–2203, 2012.
- [177] Yuan Zhou, Xi Chen, and Jian Li. Optimal PAC multiple arm identification with applications to crowdsourcing. In *ICML*, pages 217–225, 2014.
- [178] James Y Zou, Kamalika Chaudhuri, and Adam Tauman Kalai. Crowdsourcing feature discovery via adaptively chosen comparisons. *HCOMP*, 2015.

