PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Crowd and Hybrid Algorithms for Cost-Aware Classification

Evgeny Krivosheev

Advisor: Prof. Fabio Casati

May 2020

Abstract

Classification is a pervasive problem in research that aims at grouping items in categories according to established criteria. There are two prevalent ways to classify items of interest: i) to train and exploit machine learning (ML) algorithms or ii) to resort to human classification (via experts or crowdsourcing).

Machine Learning algorithms have been rapidly improving with an impressive performance in complex problems such as object recognition and natural language understanding. However, in many cases they cannot yet deliver the required levels of precision and recall, typically due to difficulty of the problem and (lack of) availability of sufficiently large and clean datasets.

Research in crowdsourcing has also made impressive progress in the last few years, and the crowd has been shown to perform well even in difficult tasks [Callaghan et al., 2018; Ranard et al., 2014]. However, crowdsourcing remains expensive, especially when aiming at high levels of accuracy, which often implies collecting more votes per item to make classification more robust to workers' errors.

Recently, we witness rapidly emerging the third direction of hybrid crowd-machine classification that can achieve superior performance by combining the cost-effectiveness of automatic machine classifiers with the accuracy of human judgment.

In this thesis, we focus on designing crowdsourcing strategies and hybrid crowd-machine approaches that optimize the item classification problem in terms of results and budget. We start by investigating crowd-based classification under the budget constraint with different loss implications, i. e., when false positive and false negative errors carry different harm to the task. Further, we propose and validate a probabilistic crowd classification algorithm that iteratively estimates the statistical parameters of the task and data to efficiently manage the accuracy vs. cost trade-off. We then investigate how the crowd and machines can support each other in tackling classification problems. We present and evaluate a set of hybrid strategies balancing between investing money in building machines and exploiting them jointly with crowd-based classifiers.

While analyzing our results of crowd and hybrid classification, we found it is relevant to study the problem of quality of crowd observations and their confusions as well as another promising direction of linking entities from structured and unstructured sources of data. We propose crowd and neural network grounded algorithms to cope with these challenges followed by rich evaluation on synthetic and real-world datasets.

Keywords: crowdsourcing, machine learning, human-in-the-loop, classification

Acknowledgements

Three and a half years of my Ph.D. work passed quickly, however, they were very rich and fruitful for my personality and career. First of all, I am grateful to Prof. Fabio Casati who offered me the scholarship and advised me during all the time of my Ph.D. journey. I enjoyed it a lot when we were discussing some algorithms and deriving together formulas for our papers, thinking out loud why it works and why it does not. I have learned a lot from you, how to do research, how to manage the processes, and how to communicate.

I would like to thank Siarhei Bykau who gave me the first taste of research, showed what it is to be a researcher, and encouraged me to start Ph.D. at the University of Trento. I am grateful Dimitri Kirov for his support, advice, and friendship from the very beginning of my Ph.D., actually from my first step in Trento, and to this day.

My **parents**, thank you for helping me and supporting with advice you could give at your best. You are the best parents and your love and care cannot be measured. My love **Svetlana**, thank you for being with me in all our life adventures and for believing in me sometimes even more than me in myself. Thank you for your sincere love, it encourages me to move forward and reach new frontiers.

Evgeny K.
Trento, 7 May 2020

Contents

Chapter 1

Introduction

Classification is a pervasive problem in research that founds a lot of applications in a variety of real-world tasks, such as document categorization, image segmentation, medical and governmental decisions. Because of their importance and impact, the number of published papers is rapidly growing, especially in medical domain (twice as many articles as in computer science field). Thus, in 2011 the number of classification related papers held $35k$ across all the fields, while in 2018 this number increased to $63k$ papers (according to our analyses conducted over Scopus[1] database).

The problem of classification can be solved via a plentiful list of methods, e. g., experts, machines, crowdsourcing, or hybrid approaches that combine a variety of different techniques [Pratap Chandra Sen, 2020; Jing Zhang, 2016; Cheng and Bernstein, 2015]. All of the methods have their benefits and disadvantages. For instance, expert classification produces accurate results but hiring experts is expansive. On the other hand, machine learning classifiers are scalable and potentially cheap once trained, but require advisory training data to rely on, and it is not always easy or even possible to train accurate classifiers for a given problem.

Crowdsourcing is a method for outsourcing small pieces of work to a crowd of people and leveraging their wealth of knowledge to achieve objectives of requesters. Crowd classification has been studied for centuries, at least since the early work of the Marquis de Condorcet who, trying to make a case for democracy, proposed his *Jury Theorem*[2], stating that if each juror in a jury (or each citizen, when taking a binary decision) has an error rate lower than 0.5 and if *guilty* vs. *innocent* votes are independent, larger juries reach more accurate results, and approach perfection as the jury grows. Nowadays, crowdsourcing acts via online crowd marketplaces (such as Figure Eight, Amazon Mechanical Turk, or Yandex Toloka), where people can perform small micro-tasks as the distributed and independent

[1] https://www.scopus.com/

[2] http://www.stat.berkeley.edu/ mossel/teach/ SocialChoiceNetworks10/ScribeAug31.pdf

crowd of workers. The workers can be motivated by receiving monetary rewards after completing a task or by acting as volunteers. Recently, crowdsourcing is being increasingly adopted as a tool for supporting research [Law et al., 2017]. There are hundreds of citizen science projects that leverage crowdsourcing at one phase or another of the research, in all fields of science, from biology to classifying galaxies [Simpson et al., 2014] to helping doctors to determine whether a treatment works[3] and many others. With the opportunity offered by crowdsourcing on the one hand and the need for large labeled dataset to support machine learning tasks on the other, many researchers from the machine learning, database, and human computation communities investigated the problem in detail, both to identify algorithms and to study theoretical bounds.

In spite of rapid technical advances in *machine learning*, nowadays many intelligent tasks still cannot be entirely assigned to machines without human supervision. There is a spectrum of problems where machines are limited under practical conditions, however, they can be assisted by humans to mitigate current limitations. Artificial Intelligent models can suffer from *low data quality* as datasets in the real-world are often incomplete, having missing values, and typos. Recent works suggest that we can combine machine-based data cleaning under real crowd supervision [Chu et al., 2016] or reassess already labeled training samples via humans [Lin et al., 2016], growing performance of models. For many real-world problems, we might possess a little or zero training data while having a large set of unlabeled items at hand. The *lack of training data* can be mitigated by combining crowdsourcing and active learning techniques that look for the most informative items and accurate workers, thus saving the budget [Heinecke and Reyzin, 2019]. Another significant challenge in machine learning is the *limited reliability of a machine's decision.* When machines are treated as black boxes without human supervision, users might hesitate about decisions and request justification and further actions. This is especially true in essential domains, such as medicine, security control, or in scientific reviews. However, such models can be assisted by humans through validation, correction, and refinement of a current algorithm [Wang et al., 2019; Kulesza et al., 2015; Ribeiro et al., 2016].

In this thesis, we direct our attention to classification problems and how we can efficiently approach it with the use of the crowd and machine learning under budget constraints. Therefore, a requester can efficiently classify a set of items specifying the available budget and desired quality requirements. This is a challenging task for many reasons: from the possible low quality of workers to data management and task design from crowdsourcing point of view; the need of obtaining and training the machine classifier from machine learning side; and challenges of how to join the aforementioned forces (crowd and machines) to increase the classification performance and reduce the budget. Therefore,

[3] https://crowd.cochrane.org/

the *objective of this thesis is to design crowdsourcing strategies and hybrid crowd-machine approaches that optimize the item classification problem in terms of results and budget.* In the following, to explain the concepts we will use the example of Systematic Literature Reviews (SLRs) - reviews that follow a predefined process aimed at achieving transparency and impartiality with respect to the sources analyzed, minimizing distortions, biases, and conflicts of interest [Grant and Booth, 2009; Khan et al., 2003; Henderson et al., 2010]. Particularly, we target the screening phase of SLRs, where we screen abstracts of candidate papers resulting from the initial literature search to identify papers to be included in the analysis. This is an instance of *finite pool* classification problems, where we need to classify a finite set of items minimizing cost [Nguyen et al., 2015a]. It is, however, rather challenging for many reasons and very important task, since SLRs are one of the most important forms of publications in science [Sun et al., 2016], and are the basis for evidence-based practices and government policies, from education to healthcare, as they pool results independently obtained from a number of research groups [Haidich, 2010]. Therefore, this task requires (i) identifying the feasibility of crowdsourcing for item classification (in particular facilitating SLRs); (ii) derive and analyse a set of crowd-based strategies; (iii) determine how artificial intelligence (AI) and crowdsourcing can be used in combination to support cost-aware item classification. We envision that AI may help in (at least) three ways that can be used independently or in combination: i) by classifying papers (acting as a cheap classifier), ii) by assisting crowd workers in performing their tasks faster and/or more accurately, and iii) by driving the crowdsourcing process, determining which item should be classified next as well as by how many workers (more accurate workers are often more expensive).

Given a budget and a set of items we might encounter some very difficult samples for crowd classification. For such difficult items, we can end up with an optimal strategy to leave them as "unclassified". Intuitively, it means we pay for domain experts to classify difficult items, and it simply costs us more money while not sacrificing the quality of results. It is essential to address this aspect especially if we have different loss values for false positive and false negative types of errors, i. e., when harm to the task varies for different wrong classification decisions. For instance, in medical tests, the loss value (or harm) for a false negative diagnosis decision should be higher than the loss for a false positive one. This scenario is common in many domains, such as security control, medical treatment, or SLRs, where missing relevant papers can lead to a wrong conclusion. It brings to us another challenge of identifying for what item we can or cannot use the crowd annotations. This is very task and question specific, as even for the same SLR question we can have different "good" and "optimal" strategies depending on the data and crowd workers.

Specifically, we investigate the following research questions:

RQ1. How can we create data adaptive crowdsourcing strategies for optimization quality vs. cost trade-off in classification with a different loss of errors?

RQ2. How to build efficient hybrid crowd-machine classifiers under a limited budget?

While exploring RQ1 and RQ2, we encountered the challenge of managing noisy crowd data and found that subtle form of crowd errors is due to *confusion* of observations on items with similar class labels. Therefore, reducing crowd noise can improve classification accuracy, and if we can decrease the number of confused crowd observations, during the crowdsourcing process, it will save the requester's budget due to collecting less crowd data for performing accurate classification. To this end, we studied crowd aggregation, confusion prevention and correction methods by investigating the third research question:

RQ3. Can we improve the quality of observations in crowd-based classification?

We further studied the issue of linking entities from structured and unstructured data sources with the use of Graph Neural Networks. We first investigate how the entity linkage can be done by machines only and then recommend a way to improve the performance with the infusion of crowdsourcing in the process. Thus, for facilitating the search of relevant literature in SLRs, we potentially can build a linkage system to perform a semantic search of similar papers based on scientific abstracts or textual queries as input. We formulate the following research question:

RQ4. How to efficiently link entities from structured and unstructured data sources?

1.1 Thesis structure

This thesis is organized as a summary of the work carried out during the three years of the Ph.D. program at the University of Trento. The core of the work is the optimization of accuracy vs. cost trade-off in finite poll classification, prevalently focused on paper screening in SLRs. First, we motivate and discuss the problem of SLRs and study the feasibility of the crowd to perform SLR tasks as well as proposing cost-efficient techniques for crowd-based classification. Then we study what machine classifiers can gain from crowdsourcing means, and how, at the same time, the crowd can benefit from machines in return. We propose algorithms for aggregating and correcting crowd votes, where the observed labels can be confused during the measurement of value. While the primary focus of this Ph.D. is the optimization of finite pool item classification, we also contributed to the graph neural entity matching problem, with a potential crowd contribution in future work. The proposed algorithms and strategies are validated on both synthetic and real-world datasets. Most of the work conducted is published (or under review) at the top conferences in the field. The graph neural entity matching algorithm is implemented as a working

application. Part of the results of this Ph.D. is based on productive collaboration with Delft University of Technology, The University of New South Wales, Purdue University, and two internships at IBM Research and IBM Benelux.

Chapter 2 discusses challenges in Systematic Literature Reviews and motivates the importance of the screening phase of reviews. We explore the feasibility of crowdsourcing in performing scope-based classification of papers in literature reviews in terms of accuracy, time, and budget. In this chapter, we propose a set of crowdsourcing techniques for item classification based on the budget available and different loss for errors (RQ1) [Krivosheev et al., 2017].

Chapter 3 aims at investigating multi-predicate classification, i. e., where we look for items that satisfy a conjunction of predicates within a finite pool of candidate items. As a result, we present the probabilistic crowdsourcing model that iteratively learns statistical parameters of the problem in order to minimize the number of crowd votes to query to reach a classification decision. The model also decides whether an item is difficult and it is better to assign it to an expert rather than spending money with the crowd (RQ1) [Krivosheev et al., 2018b].

Chapter 4 explores how to join the crowd and machine forces to classify items that satisfy a set of predicates. We show how, given a new classification problem and a set of classifiers of unknown accuracy for the problem at hand, we can identify how to manage the accuracy vs. cost trade-off by progressively determining if we should spend budget to obtain test data (to assess the accuracy of the given classifiers), or whether we should leverage the existing machine classifiers with the crowd, and in this case how to efficiently combine them based on their estimated characteristics to obtain the classification (RQ2) [Krivosheev et al., 2018a,c].

Chapter 5. In this chapter, we contribute a formulation of the general problem of hybrid crowd-machine classification, particularly in the case where no pre-trained machine learning model is available. We present and evaluate a set of budget allocation heuristics for balancing between *learning* machines (investing money in collecting training data) and *exploitation* the learned model by applying hybrid classification using the remained budget. Thus aiming at identifying how to balance the learning vs. exploitation trade-off to improve the overall classification cost and quality (RQ2) [To be submitted, EMNLP/HCOMP 2020].

Chapter 6. In this chapter, we demonstrate another reason for noise in crowdsourced annotations that leads to low-quality inference of true labels. In particular, we introduce the notion of *confusion of observations*, i. e., where crowd workers can confuse items of a class i with items of a class j, either because they are similar or because the task description has failed to explain the differences. We show that confusion of observations

can be a frequent occurrence in many tasks and that such confusions induce significant noise in datasets. We then propose an algorithm for detecting such confusions, leveraging an inference procedure based on Markov Chain Monte Carlo (MCMC) sampling (RQ3) [Under review VLDB 2020].

Chapter 7 presents the algorithm for modeling and matching entities from structured data, such as relational databases, as well as unstructured sources, such as free text from news articles. This is achieved by combining siamese and graph neural networks to propagate information between connected entities and support high scalability. We evaluate the algorithm on real business data and then discuss potential challenges and ways to improve the results from both neural network and crowdsourcing sides (RQ4) [Under review ICML 2020].

1.2 Contribution

This thesis contributes to cost-aware crowd and hybrid crowd-machine learning classification, aggregation of crowdsourced data, and graph entity matching algorithms. The contribution of this thesis can be summarized as the following.

We performed a number of analyses indicating that crowdsourcing literature reviews can be done with high precision and costs figures that are reasonable to what authors spend today in terms of effort. We proposed a model that provides users with a list of options and error-cost estimates for each option in crowdsourcing classification. The model gives an insight about specific values of task parameters (e. g., budget, number of votes per item, number of test questions) and expected results. Therefore, users can adjust the task parameters during the crowdsourcing process and consequently maximize the expected quality of results.

We introduced and examined a set of cost-aware crowdsourcing strategies for multi-predicate classification. Specifically, we proposed multi-run strategies and the adaptive (to the task) algorithm that iteratively learns statistical parameters of the problem. The algorithms significantly outperforms basic majority voting, expectation-maximization based approaches in terms of cost and accuracy for crowd classification tasks, and especially for literature reviews.

We proposed methods for combining machine learning classifiers and crowd workers to achieve higher classification accuracy for a unit of cost. We showed how to utilize potentially weak machine classifiers - and even classifiers with unknown accuracy for the problem at hand - in contexts characterized by very demanding requirements in terms of accuracy (as is the case for literature reviews) and in the presence of difficult problems where also the crowd has low accuracy. We also studied how the effect of incorporating

classifiers (and ensembles of classifiers) into a crowdsourcing algorithm varies based on the different characteristics of the problem at hand (such as accuracy of classifiers, correlation among base classifiers, amount of data available for testing classifiers and for training ensembles) and showed why and under which conditions ensembles of classifiers do or do not provide benefits over "simply" using the best available classifier at hand.

We proposed and discussed a set of budget allocation heuristics for efficiently balancing between learning and exploitation of machine classifiers in the hybrid crowd-machine classification task. Our results showed that even a small fraction of the budget invested in learning machine (e. g., via active learning) leads to improvement in cost and accuracy. Increasing the budget allocation to learning machines up to 50% increases accuracy but also cost, and from that point on the improvements can vary.

We introduced the notion of confusion of crowd observations, that is, where crowd workers can confuse items of a class i with items of a class j. To detect and correct such confusions, we proposed a novel inference algorithm based on Markov Chain Monte Carlo sampling. We showed how the proposed inference procedure can be integrated with the main state-of-the-art aggregation techniques, that it scales both in the number of items and in the number of crowd workers, and that it significantly outperforms commonly used aggregation methods especially when the number of votes per item is relatively small (in the single-digit range), as it is common in many crowdsourcing tasks. We also analyzed in which cases strong baselines such as D&S become competitive even in the presence of confused labels.

We proposed Siamese Graph Convolutional Network architecture for modeling and matching entities from structured data (e. g., relational databases) as well as unstructured data sources (e. g., news articles). We demonstrated that proposed entity matching network significantly outperforms both traditional rule-based systems and other deep learning approaches. Our graph-based entity matching architecture is deployed as a RESTful web service that accepts a company name as input and provides back the corresponding business entity in the database. Demo video is available online[4].

1.3 Publications

Research carried out during this Ph.D. resulted in the following publications:

1. *Balancing Learning vs. Exploitation Trade-off in Active Hybrid Classification.* To be submitted, EMNLP/HCOMP 2020.

2. *Detecting and Preventing Confused Labels in Crowdsourced Datasets.* Evgeny

[4]https://youtu.be/LDzCZNrCm3o

Krivosheev, Siarhei Bykau, Fabio Casati, Sunil Prabhakar. Under review VLDB 2020.

3. *Graph Neural Networks for Data Integration.* Evgeny Krivosheev, Mattia Atzeni, Paolo Scotton, Fabio Casati, Katsiaryna Mirylenka. Under review ICML 2020.

4. *Combining Crowd and Machines for Multi-predicate Item Screening.* Evgeny Krivosheev, Fabio Casati, Marcos Baez, Boualem Benatallah. ACM CSCW 2018.

5. *Leveraging Crowd and AI to Support the Search Phase of Literature Reviews.* Evgeny Krivosheev, Jorge Ramrez, Marcos Baez, Fabio Casati, Boualem Benatallah. AAAI HCOMP 2018.

6. *Crowd-based Multi-Predicate Screening of Papers in Literature Reviews.* Evgeny Krivosheev, Fabio Casati, Boualem Benatallah. IW3C2 TheWebConf (WWW) 2018.

7. *Crowd-Machine Collaboration for Item Screening.* Evgeny Krivosheev, Bahareh Harandizadeh, Fabio Casati, Boualem Benatallah. IW3C2 TheWebConf (WWW) 2018.

8. *CrowdRev: A platform for Crowd-based Screening of Literature Reviews.* Jorge Ramirez, Evgeny Krivosheev, Marcos Baez, Boualem Benatallah, Fabio Casat. ACM Collective Intelligence 2018.

9. *Crowdsourcing Paper Screening in Systematic Literature Reviews.* Evgeny Krivosheev, Valentina Caforio, Boualem Benatallah, Fabio Casati. AAAI HCOMP 2017.

Chapter 2

Crowdsourcing Classification Under Budget Constraints and Different Loss of Errors

In this chapter, we aim at designing crowdsourcing strategies for cost-aware classification under budget limits and different levels of harm for false positive and false negative errors. A different loss of errors is common for many tasks, such as medical diagnosis, criminal judgment, self-driving car decisions. This chapter discusses, the problem in light of classifying scientific papers for systematic literature reviews, which is a rather challenging task for non-expert crowd workers.

A literature review is a form of scientific research (and of publication) that has a high impact on science and society [Sun et al., 2016]. Reviews can take different forms and have different objectives [Grant and Booth, 2009]. The main distinction is between *systematic* approaches, where a specific process is defined before the review starts and is followed throughout the identification and analysis of relevant literature, and *non-systematic* ones, where authors do not follow a predefined method for selecting and analyzing literature.

Literature reviews, especially when systematic, provide scientific results and are at the heart of evidence-based approaches, with a potentially profound impact on society [Haidich, 2010]. Reviews are also very helpful in introducing newcomers to challenges and opportunities for research in a given area. Not surprisingly, they are among the most highly cited papers (a search we conducted over a few thousands papers on Scopus shows that the median number of citations for reviews vastly exceeds the median for papers in all areas of science).

Because of their importance and impact, the number of published reviews is rapidly growing [Wallace et al., 2013]. This is particularly true for systematic reviews and meta-

analyses, in the past popular mostly in the medical field but now widely adopted in all areas of science.

However, reviews are very time-consuming and effort-intensive. While there are no published statistics on the entire review process (from idea to publication) we are aware of, a study we are conducting with researchers from different fields points to durations of 6 months to 3 years from initial search to submission[1]. Review results should also be updated periodically, but again the effort for doing so often represents a barrier [Takwoingi et al., 2013].

In this chapter we investigate the possibility of crowdsourcing specific aspects of systematic literature reviews. We focus specifically on identifying the in-scope papers after initial literature search, and we investigate if and how this phase can be sourced from the citizens, what are the best strategies for doing so, and what is the resulting quality and cost, both in general and compared with the case where the same phase is done by the research team (typically, the co-authors). This is a critical phase of a systematic review: not only is it time-consuming (several people work on it, and the combined person-month effort is of over two months), but it is also where risk of bias lies.

More specifically, we contribute i) a probabilistic model for reasoning over the problem, for tuning the parameters of crowdsourcing tasks to minimize errors, and for providing review authors with information of budget vs error trade-offs, and ii) a set of crowdsourcing strategies and algorithms that minimize the classification error as we vary the assumptions on the model and the model parameters. Both the model and the strategies descend from experiments run on *Figure Eight*[2] and are mindful of what we can actually achieve with some of the practical constraints of typical crowdsourcing platforms. Experiments on Figure Eight are also used, in addition to theory and simulation, to validate the results as well as to derive parameters for the typical population of workers for this kind of tasks.

Last but not least, experiments provided many insights on task design, such as how the problem should be framed to increase participation and reduce errors, as well as actual pay scales considered acceptable by the community.

[1] Published data in the healthcare domain indicate that the median time from the *final* literature search to publication in a systematic review is 61 weeks - with the additional problem that over time the list of candidate papers for inclusion becomes out of date and needs to be refreshed [Sampson et al., 2008]. However the paper does not report on lag from initial search.

[2] www.figure-eight.com

2.1 Background

2.1.1 PRISMA and Systematic Reviews

Before presenting our approach we summarize methods and practices for systematic reviews. A systematic review follows a defined *process* and has transparency and clarity as its focal points throughout the whole procedure [Khan et al., 2003]. This process usually includes (i) the definition of a research question in a clear, structured and unambiguous way; (ii) the identification of all relevant papers through a search strategy that stems from the research question and specifies inclusion and exclusion criteria; (iii) the critical assessment of the included studies; (iv) the data extraction and synthesis in a standardized form, possibly with statistical analysis (meta-analysis); (v) the interpretation of the findings and exploration of any risk for bias [Khan et al., 2003; Wright et al., 2007; Harris et al., 2014; Henderson et al., 2010].

With the objective of increasing the quality of systematic reviews and meta-analyses, the PRISMA statement (*Preferred Reporting Items for Systematic Reviews and Meta-Analyses*) was devised as a guideline to help authors report their reviews in a clear and consistent way [Moher et al., 2009]. As an evolution from the QUOROM statement [Moher et al., 1999], PRISMA consists of a 27-items checklist enumerating the details to report and a flow diagram showing the phases of the selection process. Such statements are often required in any systematic review today and are essential in the medical field, where poorly reported reviews can potentially have an effect on people's health. Indeed, Clinical Practice Guidelines, i. e., "statements that include recommendations intended to optimize patient care", are "based on systematic reviews of evidence" and should "be based on an explicit and transparent process that minimizes distortions, biases, and conflicts of interest" [Steinberg et al., 2011]. Therefore, omitted details and lack of transparency can make this process difficult and contribute to low-quality, misleading guidelines.

2.1.2 Crowdsourcing and Science

Crowdsourcing is being increasingly adopted as a tool for supporting research [Law et al., 2017]. There are literally hundreds of *citizen science* projects that leverage crowdsourcing at one phase or another of the research, in all fields of science, from biology to astronomy to human sciences [Garneau et al., 2014; Swanson et al., 2015; Hennon et al., 2015; Lintott et al., 2008]. The interest in citizen science has generated a growing body of research on various aspects of the process, from understanding how researchers perceive it [H. Riesch H, 2014; Law et al., 2017], to the motivations behind citizens' participation [Eveleigh et al., 2014; Frey and Jegen, 2001], as well as process and system design [Tinati et al., 2015].

While all aspects of citizen science research are somewhat interesting and related to

this chapter, one item of particular importance is the understanding of the conditions under which researchers are motivated to (or deterred from) adopting a crowdsourcing approach. A beautiful analysis of these aspects is provided in [Law et al., 2017] who underscore that one of the concerns is related to how *reviewers* perceive crowdsourcing in research. In other words, crowdsourcing is viable if i) the authors feel that is feasible and valuable for their specific research problem and ii) the authors perceive that reviewers will find it acceptable. This is relevant because literature reviews go through peer reviews and as such the process needs to be accepted by reviewers - and by the community.

The prior art also includes several "spot" attempts at adopting some form of crowdsourcing in literature reviews. These papers provided inspiration for us although in many cases they are initial, one-off, and relatively small experiments that do not study the variations of the results of crowdsourcing tasks in terms of its content and parameters, and that in general do not have sufficient statistical power to derive conclusions and guidelines.

[Sun et al., 2016] study the feasibility of leveraging crowd workers for extracting information related to interventions from papers' abstracts in biomedical domains. The authors observe that giving more concrete examples in the instruction part can help workers be more aware of the purpose of a task. A platform for crowdsourcing narrative literature reviews is proposed by [Weiss, 2016], along with insight about challenges appearing in systematic literature reviews in new domains. [Nguyen et al., 2015a] propose an active learning approach to solving the problem of deciding whether or not a paper is relevant to a review, with a mixed human+AI approach. The authors tried to achieve maximum performance at minimal cost by intelligently choosing between crowd users and domain experts to minimize the expected loss. They performed experiments on Amazon's Mechanical Turk to classify papers from four large datasets. For crowd evaluation, classification is performed through majority voting.

[Ng et al., 2014] ran a randomized pilot study aimed at exploring the accuracy of medical students in performing citation screening via four different modalities, namely a mobile screening application, paper printed with titles and abstracts, a reference management software and a web-based systematic review platform. Students were asked to say whether a list of papers were included or excluded from the scope of a review, based on a list of inclusion criteria. In case of insufficient information, participants could set papers as "unsure". Participants had never conducted a review, however they had some level of expertise in the field and had received some training in the development of critical appraisal skills, which differs widely from asking to a non-expert crowd to perform such a task.

2.1.3 Modeling and Classification

Extensive research, dating back to the 1700s, has addressed the problem of eliciting reliable labels from a crowd, coping with cheating behavior while keeping costs low [Karger et al., 2011b; Whitehill et al., 2009; Smyth et al., 1995; Karger et al., 2011a; Hirth et al., 2013; Liu et al., 2013; Eickhoff and de Vries, 2013; Hirth et al., 2011].

One of the first scientists to study this was the Marquis of Condorcet. Condorcet, in his famous *Jury Theorem*[3] of 1785, discusses the probability of a group of persons taking, collectively, the correct classification decision. He shows that if the probability of a person taking the correct decision is greater than 0.5 and votes are independent of each other, then the probability of taking a correct majority decision grows with the number of participants and approaches 1 at the limit (this is, in fact, a direct consequence of the law of large numbers).

From there, a large body of work starting with [Dawid and Skene, 1979] and then on to [Whitehill et al., 2009], estimating labeling in the presence of items of different difficulties, and [Liu and Wang, 2012] who apply EM to labeling in the presence of confusion matrix inspire our approach. We also build on insights from [Hirth et al., 2013], who discuss the problem of cost optimization providing information on which cheating detection and task validation algorithms to choose based on the cost structure of the task. Our work differs in that we seek for a method to provide, for each task, review authors with a description of price vs error trade-off, an optimal choice of parameters for a given price, and a set of crowdsourcing strategies that aim at minimizing error estimates. In other words, we don't "simply" aim at classifying papers given the votes from the crowd, but we identify the strategies to obtain such votes by considering the price vs error trade-offs.

2.2 Model and Assumptions

2.2.1 Task Model

We consider a crowdsourcing task model that includes set of candidate papers $CP = \{p_1, p_2, ..., p_n\}$ and a textual definition of the scope of the review S. The task is performed by workers in a pool of contributors. In practice this pool is very large and for our purposes we assume it is infinite. We then ask each worker j to label one or more candidate papers as *in* (the paper is in scope or we do not have sufficient evidence to exclude it from the abstract and title) or *out*, based on S. In case of exclusion, workers are asked to provide reason to do so. Figure 2.1 shows an example task for a review we recently completed.

The result of a task execution is a set of votes $L = \{l_{pj}\}$ representing the binary vote

[3] http://www.stat.berkeley.edu/ mossel/teach/ SocialChoiceNetworks10/ScribeAug31.pdf

Sign	**Specification**
$CP = \{p_1, ..., p_n\}$	set of candidate papers
l_{pj}	vote by worker j on paper p
cr	cost ratio
N_t	#test questions
N_l	#votes per worker (after passing test)
J	#votes per paper
S	scope of a review
UC	cost per vote
PPP	price per paper
z	proportion of cheaters
$\overline{a}_s$	expected accuracy of workers
θ_i	real proportion of papers *in* scope

Table 2.1: Notations used in the chapter.

of worker j on paper p. Given the set of votes, we use a classification function $cls(L)$ that takes the individual votes and aggregates them to derive the in/out decision for each paper.

Finally, we define the costs (*loss*) for each error: a cost for a false exclusion $Cost_{fe}$ (deciding that a paper is out when it should have been in), and for a false inclusion $Cost_{fi}$. For simplicity we model them as a cost ratio cr that defines how more costly a false exclusion is with respect to a false inclusion. False exclusions are typically more costly since we may be missing an important paper, while a false inclusion "simply" means that experts will need to go over that paper again. The value of cr depends on the subjective opinion of review authors.

A run of a crowdsourcing task proceeds as follows[4]: first each worker is shown a set N_t of test questions with known labels. If the worker answers them correctly, they move to the work phase, where they can provide useful votes (that is, label unknown candidate papers). Even during the work phase, test questions may be injected with a defined frequency and the contributor is considered trusted (their results are not discarded) only if they keep answering the test questions correctly. The run continues until a given number of labels per paper J has been reached. We assume test questions are created by authors based on the problem at hand, that is, based on a screening they perform over a handful of

[4]The choice of the model is also guided by what we can do today with platforms such as Figure Eight

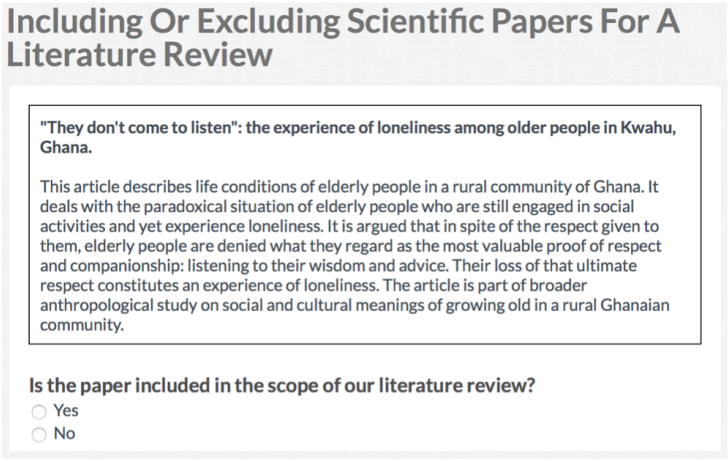


Figure 2.1: Example of scope-based screening task.

papers. Typically, creating ten test items that include inclusion and exclusion examples is sufficient for crowdsourcing purposes.

In the simplest case a task will have just one run, but we can envision that a run may leave us with uncertainty over some papers and we may want to have additional runs focused on uncertain papers.

Last but not least, each task has a *price*. The price tag is affected by: i) the unit cost per label (how much we pay workers for labeling a paper or an exclusion criterion), ii) the total number of votes asked, and iii) the number of test questions. The first two are rather obvious, while the third requires an explanation: with infinite workers, to get accurate results we might simply have a very large number of test questions so that we are sure that only trusted, competent workers remain in the task. In many systems test questions are not paid, so this costs zero. In practice this is not possible: the ethics of this are questionable at best, non-cheaters would do a lot of unpaid work, and we, as task providers, would get bad ratings, impacting our future ability to crowdsource. In this chapter, we take this into account by increasing the price per judgment by a factor $\frac{N_l+N_t}{N_l}$, where N_t is the number of initial test questions and N_l is the number of valid judgments that a worker gives on non-test papers (i.e., the number of votes from a worker who remains above the threshold tr). This essentially states that people who pass the test are in fact paid also for test questions. As N_l grows our factor becomes ineffective and others can be chosen, but in our case N_l is small[5]. Alternative models can be derived, also including a penalty for high test frequencies, but for presenting the concepts and ideas of

[5]In practice, depending on the task settings, it may not always easy to enable a worker to label many papers due to the fact that many concurrent workers access the task in parallel and the available work finishes very quickly.

this chapter this is sufficient. The classification cost for a paper is therefore expressed as follows, where US is the cost per vote and $\frac{N_l+N_t}{N_l}$ is the corrective factor:

$$PPP = UC \cdot J \cdot \frac{N_l + N_t}{N_l} \tag{2.1}$$

In the end we want to perform candidate paper selection with high accuracy (minimizing the loss) and minimal price. A specific point of interest lies in whether the crowd can achieve an accuracy similar to (or better than) that of experts at a comparable cost, while ensuring transparency and impartiality of the whole process.

2.2.2 Probabilistic Model

To reason about the model and identify strategies and parameters we define a probabilistic model that describes the characteristics of i) tasks and ii) workers. Both come into play to identify the optimal crowdsourcing strategy and to set the crowdsourcing parameters.

With respect to the task, we model the following:

1. Our *belief* on the proportion of candidate papers that should be included. This is important because it affects the classification function. We do not assume that authors necessarily have such a prior belief, and we discuss later how this parameter can be set or estimated.

2. The *difficulty* level of each paper: we need to account for the fact that not all candidate papers are equal, meaning that some papers may be harder than others to classify. In this chapter, we model difficulty with a uniform distribution (which we can parametrize with a variety of priors, such as the commonly used $Beta(\alpha, \beta)$ or priors as suggested in [Whitehill et al., 2009]).

With respect to the workers, we assume that in the worst case workers answer randomly, which means a 0.5 probability of a correct label. The proportion of cheaters is modeled by a Bernoulli random variable Z. For non-cheaters, we initially assume a uniform accuracy from 0.5 to 1. The accuracy probability function is therefore a mixture of a point mass at 0.5 and a density in the (0.5, 1) range.

$$pdf(a) = z \cdot \delta(a - 0.5) + 2 \cdot (1 - z) \tag{2.2}$$

for $0.5 \leq a < 1$. In the function, δ is the impulse function, while the uniform density is multiplied by 2 (as it is in the $(0.5, 1)$ interval only) and by $(1 - z)$ as the density applies only to non-cheaters. In this chapter, we do not include more complex cases that include a confusion matrix or priors on the initial probability, but the concepts can be extended to that case.

2.3 Calculating error cost and price

Now that we have a model we can reason about *strategies* for crowdsourcing literature reviews and assess them based on *assumptions* we can make related to the model.

The goal is to i) identify which aspects of the model impact the selection of strategies and results, ii) estimate the model parameters (or at least refine our prior, when available) based on actual experimental data, and iii) derive which strategies can lead to good results in terms of error cost (loss) and price. Because each problem is different (and even varies also depending on how we title or present the task, as discussed later), the statistical parameters will also vary, so while we can inform our priors via experiments, each task may have to refine the estimation on the go.

We begin by studying a simple version of the model and a simple crowdsourcing *strategy*. In general, a crowdsourcing task for literature review can be comprised of a number of *runs*, where in each run k we submit a subset CP_k of the candidate papers CP to the crowd, collecting a given number J^k of labels per paper. Furthermore, we start each run with a belief B^k on the proportion of papers to be included, if available (and initially assumed to be 0.5 if there is no estimate). A run R^k is therefore defined by a tuple (CP^k, N_t^k, J^k, B^k).

In the simplest strategy the task consists of one run where we submit all papers and seek for J votes per paper. A classification function will then classify the paper based on the cost ratio cr, trying to minimize the loss while fitting within an experiment budget.

The objective for the algorithm here, before even proceeding with the classification, is to i) estimate the optimal values for task parameters that we (as managers of the crowdsourcing process) can play with, such as number of test questions N_t, the requested judgments per paper J, and the classification function, and ii) provide the scientists with a *budget vs expected loss* curve, showing the error cost depending on the budget, assuming that for each budget we choose the best (lowest loss) configuration identified. The only input explicitly required by the authors is the cost ratio, which is subjective.

The expected error cost (loss) for each paper is given by formula 2.3, where P(FE) and P(FI) denote the probability of false exclusion and false inclusion.

$$Loss = cr \cdot P(FE) + P(FI) \tag{2.3}$$

Considering that we obtain J judgments (votes) per paper, if we decide to exclude a paper after we obtain J_t exclusion votes or more for such a paper, the probability of a false exclusion is given by equation 2.4, where θ_i is the (initially unknown) probability that the correct decision for a paper is inclusion, and $\overline{a}_s$ represent the expected accuracy of workers who pass the test phase. The formulas descend from the observation that P(FE) $= P(decision = exclude/correct = include) \cdot P(correct = include)$, and vice versa for

P(FI).

$$P(FE) = \theta_i \cdot \sum_{J_t \leq k \leq J} \binom{J}{k} \cdot (1 - \overline{a}_s)^k \cdot \overline{a}_s^{J-k} \tag{2.4}$$

$$P(FI) = (1 - \theta_i) \cdot \sum_{J - J_t < k \leq J} \binom{J}{k} (1 - \overline{a}_s)^k \overline{a}_s^{J-k} \tag{2.5}$$

In this formula, θ_i is an unknown parameter we need to estimate, $\overline{a}_s$ is also an unknown parameter on which, however, we can have some control by adjusting the test questions N_t to filter inaccurate workers, while J and J_t can be set to optimize loss. The accuracy $\overline{a}_s$ of the population that survives N_t tests is distributed as follows: if we denote with z_s the proportion of cheaters in the population that survives the test, which can be derived from Bayes ($z_s = P(test\_passed/cheater) * P(cheater)/P(test\_passed)$), then

$$Z_s = \frac{z \cdot 0.5^{N_t}}{(z \cdot 0.5^{N_t}) + (1 - z)\frac{2}{N_t+1} \cdot (1 - \frac{1}{2^{N_t}+1})} \tag{2.6}$$

Consequently, by using again Bayes for deriving how the accuracy of non cheaters, initially uniform, is reshaped by the test questions, we obtain:

$$f^t(a) = z_s \cdot \delta(a - 0.5) + (1 - z_s)\frac{2^{(N_t+1)} \cdot (N_t + 1)}{2^{N_t+1} - 1} \cdot a^{N_t} \tag{2.7}$$

for $0.5 \leq a < 1$

The expected accuracy $\overline{a}_s$ of this population is $E[x] = \int_{0.5}^{1} x \cdot f^t(x)dx$ and is therefore shown in Equation 2.8.

$$\overline{a}_s = \frac{z_s}{2} + (1 - z_s)\frac{2^{(N_t+1)} \cdot (N_t + 1)}{2^{N_t+1} - 1}(\frac{1 - (0.5)^{N_t+2}}{N_t + 2}) \tag{2.8}$$

2.4 Error minimization and error/price trade-offs

We begin our discussion on algorithms by assuming a single-run strategy.

2.4.1 Single-run strategy with simple majority voting.

In this approach we simply classify papers using majority voting, which is the approach most commonly supported by crowdsourcing platforms. For each combination of N_t, N_l and J we can compute the total price tag of the experiment as well as estimate the loss via equation 2.3, where J_t is set to $J/2$ (rounded to the upper integer), as shown in Figure 2.2. As we have no knowledge of θ_i, we assume a value (such as 0.5, though different values can be set if the task requester has a prior belief). In practice, values of N_t and J over 10

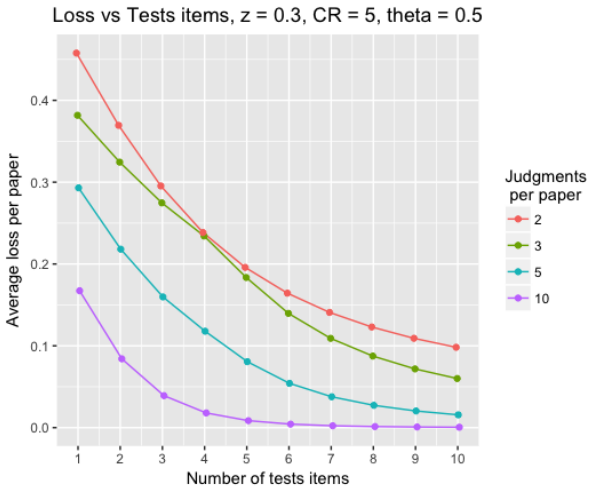


Figure 2.2: Expected loss depending on the number of test questions and of judgments per paper (real $\theta_i = 0.5$).

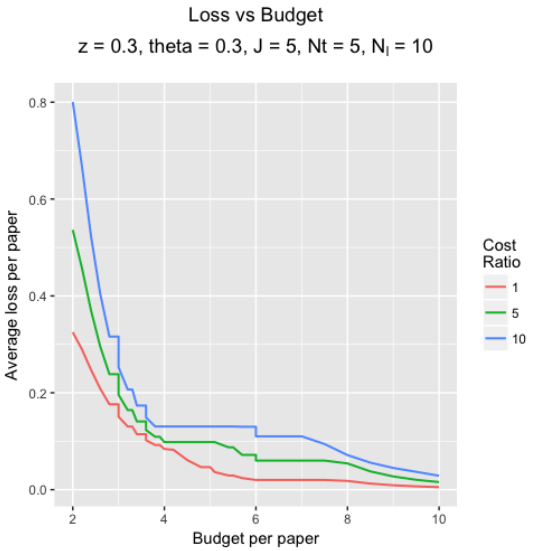


Figure 2.3: Expected loss that can be achieved depending on the budget (real $\theta_i = 0.3$).

result in near-zero error cost, so computing loss for higher values can be easily done but is rarely needed.

The result can be plotted as done in Figure 2.3. The decision of the optimal price/loss point is left to the user as it depends on subjective considerations as well as available budget. Each budget corresponds to an optimal choice of N_t, N_l and J that fits in the budget with minimal loss, so that once we have the requestor's decision we can configure the crowdsourcing task. Notice that for now we are assuming that our initial guess of θ_i is correct.

We can then classify the paper using majority voting. Figure 2.4 shows the performance of this algorithm (denoted by MV in the legend) in terms of expected loss, assuming an initial run with five tests. Figures 2.4(b), top and bottom, differ from Figures 2.4(a) as we assume a more difficult set of papers, in this case simulated by scaling down the accuracy of non-cheaters to the 0.5-0.7 range. Furthermore, the top charts have a lower values for J and cr. The increase with θ_i here is due to the fact that this method does not consider the cost ratio which typically penalizes false exclusions. Therefore, error cost grows with the proportion of included papers.

2.4.2 Single-run strategy considering cost ratio.

The obvious improvement to the baseline is to consider the cost ratio. This time, for each value of N_t and J we can minimize the loss (according to equation 2.3) by selecting the

optimal value for J_t. Again, here we only "guess" a θ_i or set it based on the requester belief (in the following chart we assume an initial belief of 0.5). The minimization can be done using classical minimization algorithms [Arora, 2015] but also by computing the values given that we have a small number of discrete variables. For each combination we have again a price point and we can plot again loss vs price chart, ask the user to point to an acceptable compromise, determine the parameters and run the task as for the previous case.

As we can see from the results in Figure 2.4(a) (the label for this algorithm is SCR), this algorithm performs better for high values of θ_i. For $\theta_i = 0.5$ all algorithms behave similarly as the initial assumption of $\theta_i = 0.5$ holds, while for low value of θ_i the loss is higher. This is because we tend to err on the side of inclusion, so for low values of θ_i we get higher errors. However for difficult papers where the accuracy is very low, the error actually grows with θ_i, because the probability of false exclusion goes up and if workers are not precise and we do many errors, we pay a price which is not compensated by erring on the side of inclusion.

2.4.3 Single-run strategy with basic parameters estimation.

The value of the parameters θ_i and z_s plays a role in the loss function, and the cost ratio is also important for determining the optimal classification function given the outcome of a run (in our case, for determining J_t which is the only parameter left to play with once we have concluded a run). Therefore, we assume we can improve on the above method by estimating θ_i. There are many ways in which this can be done. One option is to again use *majority voting* but only for performing an initial classification. Based on this, we compute the proportion of included papers and take this as an estimate for θ_i, more informed than an initial guess of 0.5. We then compute the accuracy of each worker (as percentage of "correct" answer based on majority voting classification), and with estimates of θ_i and $\overline{a}_s$, we then compute the optimal value for J_t based on equation 2.3, and correspondingly we know the minimal loss we can achieve for each price.

As we can see from the results in Fig 2.4(a) (the label for this algorithm is BPE), this algorithm performs significantly better than the previous ones for all values of θ_i except 0.5 (where the guess of the simpler algorithms is correct).

2.4.4 Single-run strategy with EM-based parameters estimation.

We can improve on the above algorithm by iterating over estimates of the parameters until convergence. A common method for doing so is to leverage Expectation Maximization [Dempster et al., 1977; Dawid and Skene, 1979]. In our model, the data is presented

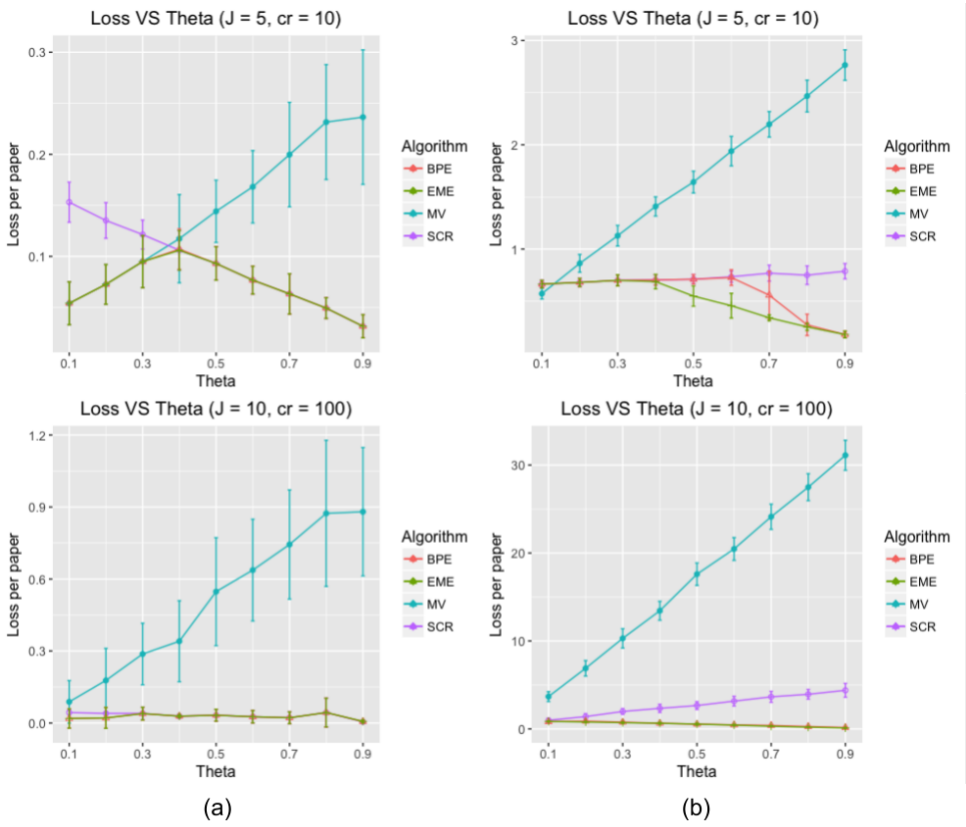


Figure 2.4: Expected loss for each algorithm. With no difficulty bias (a) and with difficulty bias reducing worker accuracy to a 0.5-0.7 range (b). MV is majority voting, SCR is Simple strategy with cost ratio, BPE is basic parameter estimation, and EME is expectation maximization. The simulation is based on 1000 papers, $N_t = 5, N_l = 10, z = 0.3$.

as a Bayesian network, where there are two types of variables: 1) the observable votes provided by workers, and 2) hidden variables, such as θ_i, the workers' accuracy, and the classification for each paper. Via the EM algorithm we computes the correctness of values given the accuracies of the workers that support it. See [Jeff Pasternack, 2011] for the details and examples of EM-based for data aggregation. The results shown in Figure 2.4 indicates that EM is equal to basic estimation and slightly better when accuracy is low.

2.4.5 Multi-run Strategies.

The big limitation in all of the previous algorithms is that we run the crowdsourcing task "in the dark". We "guess" the value of the parameters and, based on this, set the number of tests and of judgments, leaving the optimization to the post-task analysis phase, when the money has been already spent. We can improve on this by running a small test-run whose

purpose is to obtain initial estimates for θ_i and z_s. Once we get initial estimates, we can compute and plot again the budget vs loss chart, and based on the estimates and within the confines of the budget, minimize the loss, but this time with the ability to modify N_t and J based on the estimates. We call this a *horizontal multi-run strategy* as we cut the list of papers horizontally. The approach assumes that the initial sample of papers is representative of the whole set, and in absence of specific knowledge this means that we randomly reshuffle the papers before selecting the initial P papers for the estimation run.

The results are shown in Figure 2.5, depicting the price per paper we can obtain with a multirun strategy that has the same loss of a single run strategy, run with a budget of 7.5, and optimized with EM-based algorithm. We can see that for values of θ_i close to 0 or 1 a multirun strategy obtains savings of approximately 20%.

Multi-run strategies are particularly important when the difficulty of the task is unclear: the difficulty affects the accuracy as pointed out in [Whitehill et al., 2009], so that papers in certain areas may get lower accuracy than others. Similarly, we can apply a *vertical* multi-run strategy where we collect *one* vote on *all* papers, and use this to estimate the parameters, and proceed with collecting a second vote, and so on.

θ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
PPP M	6.15	6.15	6.15	6.15	7.5	7.5	7.5	6.15	6.15

Figure 2.5: Price per paper with a multirun strategy that has the same loss of a single run strategy at a price of 7.5, optimized with EM-based algorithm. Shown for different values of θ_i. Estimation based on $N_t = 5, cr = 10, z = 0.3$, run of 1000 papers.

2.5 Analysis and Experiments

In the winter and spring of 2017 we run a series of experiments on Figure Eight to assess our results and estimate parameters based on actual crowdsourcing scenarios, as well as to understand how such a task can be framed and how sensitive it is to how we word the question or to the difficulty of the papers.

We ran a total of 16 experiments with different settings, asking workers to label a total of 50 papers taken from two systematic reviews, one done by us in an area across computer science and social sciences reviewing technology for fighting loneliness, using fairly common terminology, and the other in medicine [Veronese et al., 2017] having more complex exclusion criteria, with 26 and 24 papers respectively [6]. We collected votes by 2896 respondents (807 of which passed the test phases). The price of each label per paper

[6]The detailed description of all experiments is available at https://tinyurl.com/csexphc

was also experimented, ranging from 0.22$ to 0.35$, which corresponded to approximately 10 to 15 $/hour. The purpose of the run was not so much to use Figure Eight to get the results, but to understand the workers response in terms of accuracy and speed on real tasks.

The first observation is that the price point is considered acceptable by workers. Overall, the job was rated from 3.3 to 4 in a 5-point scale, and we understand from Figure Eight that this is above average. Interestingly, there is a high variance so that sometimes a lower pay resulted in higher rating for two different tasks with the same settings. Classifications based on exclusion criteria generally get higher ratings for the same pay. On average the tasks attracted one worker every 20 seconds. Because of the large pool of workers that end up working concurrently, each worker cannot rate a high number of papers simply because we quickly reach the desired number of votes per paper.

Another observation is that the worker accuracy changes a lot depending on the subject area. The paper in the medical area, which included complex criteria for determining scope or exclusion obtained an average accuracy of 59%, versus 83% for the technology paper. Interestingly, the accuracy depends on the title we give to the task ("classify a text" vs "screen scientific papers"), probably as titles that convey that the task is complex tend to discourage the casual worker, and we know that workers correctly perceive task complexity [Yang et al., 2016]. If we word tasks properly and the problem is sufficiently simple, then as shown by Equation 2.6 the average accuracy after just a few test questions is very high, and classification errors, even using simple majority voting, are low. In this case the classification can be very precise and indeed, in our experiments, in half of the cases (4) where we recorded an "error" from the crowd, the error was on our side meaning that our "gold" data turned out to be not so gold.

We can use the accuracy distributions as derived from Figure Eight and feed them to the algorithms described earlier to compute task settings for relatively easy and relatively hard paper classification problems, and estimate loss for, e. g., a maximum budget of 1$ per paper and a salary of 20 cents per answer. For the medical domain case, the optimal algorithm produces N_t=10 and J=2, giving a cost per paper of 80 cents and an expected error loss for $cr = 10$ of 0.15 if $\theta_i = 0.5$. For reviews where the real $\theta_i = 0.1$ the loss is 0.08 for a cost of 1$ per paper (optimal parameters are $N_t = 7$ and $J = 3$). For the ("easier") technology review we can instead reach a loss of 0.11 when the real $\theta_i = 0.5$ (cost of 80cents per paper, $N_t = 10, J = 2$) and for real $\theta_i = 0.1$ the loss is 0.08 for a cost of $N_t = 6, J = 3$, budget of 96 cents per paper.

Notice that 80 cents per paper is a reasonable figure (also obtained with generous assumptions on the time actually spent working on each paper - in practice it is probably possible to achieve lower costs): in our preliminary survey of over 20 authors of recent

literature reviews, respondents reported an average of 1.5 person-months of effort spent in this phase. For a typical screening of approximately 1000 papers the price tag is therefore relatively low.

2.5.1 Conclusion

The analysis indicates that crowdsourcing literature reviews can be done with high precision and costs figures that are reasonable with respect to what authors spend today in terms of effort. Different algorithms can be used to identify the parameters of the crowdsourcing task and the best algorithm we identified based on a multi-run strategy significantly outperforms basic EM (with even larger margins when compared with other simpler algorithms). The work has several limitations: in this presentation we could only include a few comparisons and discussions. As the model (and real life scenarios) have many parameters, a more in depth discussion and analysis is needed. Furthermore, a deeper understanding of the accuracy as derived from Figure Eight is needed as it is affected by many "little things" (as we experienced almost by chance) such as the wording of the task title and content, the type of papers, the availability of a rich set of examples for the worker, and so on. Algorithms still have room for improvement, for example in terms of finding the optimal number of papers to consider for the initial run of the multi-run strategy. Furthermore, we have not discussed and analyzed the impact of clarity and of ongoing tests submitted to workers who pass the test phases. A detailed comparison with actual errors performed when experts decide inclusion and exclusion is also needed for a comprehensive evaluation of the approach.

Chapter 3

Crowd Adaptive Criteria-based Paper Screening in Systematic Literature Reviews

Systematic literature reviews (SLR) [Grant and Booth, 2009; Khan et al., 2003; Henderson et al., 2010] are reviews that follow a predefined process aimed at achieving transparency and impartiality with respect to the sources analyzed, minimizing distortions, biases, and conflicts of interest [Steinberg et al., 2011]. They are one of the most important form of publications in science [Sun et al., 2016], and are the basis for evidence-based practices and even government policies, from education to healthcare, as they pool results independently obtained from a number of research groups [Haidich, 2010]. Recognizing their importance, the number of systematic reviews is growing steadily, with tens of thousands of publications per year in all fields.

The cornerstone of transparency and impartiality in SLRs lies in a formalized paper selection process. This is typically formed by a stated scope and goal of the review (e. g., "study the effect of regular physical exercises on progress of dementia in older adults, focusing only on papers describing randomized controlled trials"), translated by the authors into a corresponding query (a boolean expression that includes relevant keywords) that retrieves candidate papers from a database such as Scopus. To avoid missing papers, the query tends to be inclusive, which means that it returns hundreds or thousands of results [Nguyen et al., 2015a] that are later screened by researches based on predefined *exclusion criteria* (e. g., "filter out papers that do not measure cognitive decline"), typically down to a few dozens.

While extremely useful, SLRs are very time consuming in terms of both effort and elapsed time, and this is true also for the paper screening phase [Green, 2011; Sampson

et al., 2008; Krivosheev et al., 2017]. Furthermore, with hundreds of thousands of papers written every year, SLRs rapidly become outdated [Créquit et al., 2016], and although they should be updated periodically, the effort for doing so often represents a barrier [Takwoingi et al., 2013], so that it is not uncommon for reviews to miss 30% or 40% of relevant papers [Créquit et al., 2016].

In this chapter, we continue to explore the use of crowdsourcing in the filtering phase of systematic reviews, where we screen candidate papers resulting from the initial literature search to identify papers to be included in the analysis based on a set of *exclusion criteria*. This is an instance of finite pool classification [Nguyen et al., 2015a] and crowd screening problems [Parameswaran et al., 2012] where we need to classify a finite set of objects while minimizing cost. The potential benefits of crowdsourcing here are in terms of a faster and cheaper screening (compared to screening by professionals) as well as increased transparency (process and votes can be made public if desired) and reduced risk of author bias. The crowd also brings diversity [Weiss, 2016] and, as we experienced first hand, disagreement in the crowd may signal errors or ambiguities in the definition of exclusion criteria. Research in this area is still in its infancy, although a set of recent initial efforts [Wallace et al., 2017c; Krivosheev et al., 2017; Sun et al., 2016; Mortensen et al., 2016] present very encouraging results in terms of both quality and cost reduction with respect to expert screening costs, and show feasibility of crowd-based screening in various domains, including healthcare.

In the following we present a probabilistic model suitable for the criteria-based screening of papers typical of SLRs and propose a set of strategies for crowd-based screening. Our main contribution consists in an adaptive crowdsourcing algorithm that significantly outperforms baselines. The algorithm polls the crowd in small batches and estimates, at each iteration and *for each item*, i) the criterion for which getting one more crowd vote on the paper can more efficiently lead us to a classification decision, and ii) whether we should give up trying to classify this item, recognizing that the crowd cannot efficiently reach a decision and therefore it should be left to the authors for expert screening. This also means that the algorithm is robust to papers and criteria that are overly difficult for the crowd to classify, in that it does not needlessly spend money on them.

The model is the result of many iterations and variations of experiments on commercial crowdsourcing platforms (Amazon Mechanical Turk (AMT) and *Figure Eight*[1]). We then performed additional experiments to validate the effectiveness of the strategies. While we present the results in the context of SLRs because we validated the model and findings for this case, we believe that results can be generally of interest for classification problems where items are classified via series of successive tests, as it often happens in medicine, as

[1] www.mturk.com and www.figure-eight.com

well as for finite pool classification problems and crowd-based query optimization, where the crowd evaluates predicates (analogously to our exclusion criteria) that filter a set of tuples to compute the query results.

3.1 Background

Our work builds on approaches in crowdsourcing in SLR but also more generally on works on crowd-based classification.

3.1.1 Crowdsourcing multi-criteria paper screening in Systematic Reviews.

Recently, Brown and Allison [Brown and Allison, 2014] used crowdsourcing to, among other tasks, classify 689 abstracts based on a set of criteria using AMT. Authors report agreement in 75% of the abstract, based on two raters, and a third rater is used to break the tie in case of disagreement. The chapter does not discuss optimal crowdsourcing strategies or algorithms to minimize errors, but points to the potential of crowdsourcing in analyzing literature.

Mortensen and colleagues crowdsourced paper screening [Mortensen et al., 2016] in four literature reviews, each with several criteria. Their aim was to explore feasibility and costs of crowdsourcing and they address the problem by measuring workers agreement in a set of tasks run on AMT for papers in the medical field. Their work differs from ours in that it does not propose algorithms to identify optimal crowdsourcing strategies. However, it contains interesting observations related to the importance of task design, to the cost-effectiveness of crowdsourcing even when the task is not optimized, and to the high degree of variability in workers's agreement from paper to paper and criteria to criteria (Fleiss' Kappa ranging from 0.5 to -0.03). This is consistent with our own studies (our papers are in a different scientific area) and we exploit this variability to optimize the cost/error tradeoff.

In general all papers reports positive results and complement them with insights and guidelines for task design of even for the design of a dedicated crowdsourcing platform for SLR [Weiss, 2016; Brown and Allison, 2014] as well as investigate the use of crowd for other phases of interest for SLR such as information extraction [Sun et al., 2016]. Interestingly, the only exception is represented by a study performed with medical students as screeners rather than online crowd workers, which reports rather poor accuracy [Ng et al., 2014].

From these studies we also learn that workers' accuracy vary across criteria, which points to the need of adapting to the characteristics of each SLR, criterion, and crowd. Indeed, one of the main differences of our approach lies in the ability to focus the crowd

on "low hanging fruits", that is, items and criteria that are statistically more efficient from the perspective of correctly excluding papers.

3.1.2 Crowdsourced Classification

The problem discussed here is an instance of a finite pool classification problem [Nguyen et al., 2015a] and specifically of crowdsourcing-based classification. Prior work also addresses the issue of optimizations in terms of costs for obtaining labels and techniques to reduce cheating [Smyth et al., 1995; Karger et al., 2011a; Hirth et al., 2013; Eickhoff and de Vries, 2013; Hirth et al., 2011]. For example, Hirth and colleagues [Hirth et al., 2013] recommend specific cheating detection and task validation algorithms based on the cost structure of the task.

We build over many of these approaches, and in fact we adopt prior art algorithms for estimating workers' accuracy and for assigning labels. Although classification algorithms are central to our overall problem, to a large extent they are for us a swappable component: Our goal is to, given a task design and a classification algorithm, identify how to efficiently query the crowd to minimize the number of labels needed to achieve the desired precision and recall in screening problems.

3.2 Model and Objective

We model the SLR screening problem as a set of papers (items) I to be classified by the screening phase as *included* (in scope) or excluded based on a set of exclusion criteria (predicates) $C = \{c_1, c_2, ...c_m\}$. A paper is excluded if at least one exclusion criterion applies, otherwise it is included. A typical SLR screens hundreds or thousands of papers with a handful of exclusion criteria. We focus on screening based on title and abstract, which is a classical first step screening, consistent with SLR guidelines [Moher et al., 2009].

In a crowdsourcing approach, we ask each crowd worker to look at one or more pairs (i, c) and state if exclusion criteria c applies to paper i. Following the mentioned literature, we model a worker's accuracy with a confusion matrix $A_{c,w}$ defining the probability of making correct and wrong classifications for each criterion c, thereby allowing us to model different accuracies when the true label is inclusion vs exclusion. Criteria can differ in *difficulty*. Some are easier to assess than others. Following [Whitehill et al., 2009], we model difficulty as a positive real number d_c that, given an expected accuracy α_w of a worker w, skews the accuracy as follows: $\alpha_{c,w} = 0.5 + (\alpha_w - 0.5) * e^{-d_c}$. As the difficulty d_c grows, $\alpha_{c,w}$ goes to 0.5, corresponding to random selection, which we consider to be the lowest accuracy level[2]. Each criterion also has a *power* (also called selectivity) θ_c, defined

[2]In this chapter, we do not consider the problem of accuracies below random, but we stress that they can

as the percentage of papers to which the criterion applies (and hence need to be excluded). For each SLR and criterion, both accuracy and power are unknown a priori.

We assume the adoption of a general purpose crowdsourcing system with limited control on the kind of crowd we attract but with a near infinite pool of workers. We can however test workers by providing a number N_t of test questions (with gold answers provided by SLR authors), and count as valid only votes of workers who pass the test, thereby exercising some control over worker's accuracy (at a cost, as we specify later).

A crowdsourcing *strategy* is a set K of runs, where each run R^k collects $J^k_{i,c}$ votes for criterion c on item i. A run may seek votes on all criteria and all papers, or focus on a subset (that is, $J^k_{i,c}$ might be 0 for some items).

Tasks also have a cost, which is the unit cost UC for a (non-test) vote multiplied by the number of votes obtained. Although many systems allow not to pay for test answers, consistently with [Krivosheev et al., 2017], we believe it is fair and ethical to also pay for test questions for workers who pass them. Furthermore, placing unreasonably many test questions is likely to result in low reputation scores for us and hence in our ability to crowdsource. Concretely, this translates into considering a price per label PPL as follows (N_l is the number of valid judgments that a worker gives on non-test papers):

$$PPL = UC \cdot \frac{N_l + N_t}{N_l}. \tag{3.1}$$

In terms of outcome, key measures are the precision and recall of papers to exclude. We also borrow the concept of loss function from [Krivosheev et al., 2017; Nguyen et al., 2015a] because it summarizes well the subjective perspective of the SLR authors. The $loss = lr * FE + FI$ is represented by the sum of false inclusions FI (papers that survived the screening phase but that should have been excluded instead) and false exclusions FE (filtered-out papers that should have instead been left in), where FE are weighed by a loss ratio lr denoting that false exclusion are lr times more "harmful" than false inclusion (filtering out a paper is often considered a much more serious error than a false inclusion which "simply" requires extra work by the authors). The loss ratio is the only parameter we ask the authors to set.

Many variations of the model and of loss function are possible, but these suffice for our purposes. Given the model, our objective is to identify and evaluate a set of efficient crowdsourcing strategies for each SLR that correspond to estimated pareto-optimal price/loss curves. With infinite money we can always arrive at a perfect classification (if workers' accuracy is above random and votes are independent), but the challenge is to classify efficiently and at a price/loss point that is acceptable to authors, who decide what price they are willing to pay and which loss they can tolerate. Based on this preference, the

occur in rare cases, for example if criteria are erroneously specified.

algorithm should set the relevant parameters of the crowdsourcing tasks and classification function. We next discuss how this can be done.

3.3 Algorithms

3.3.1 Baseline single-run algorithms

Our set of baseline algorithms follows the methods applied in recent literature for crowdsourced classification in finite pool contexts and SLRs in particular. Specifically, as we are in the presence of incomplete information (we know neither the classification of the papers nor the accuracy of the workers), we leverage approaches such as TruthFinder [Dong et al., 2013] and Expectation Maximization (EM, [Dawid and Skene, 1979]) to iteratively refine estimates of accuracy and class until convergence. In addition, simple majority voting is also commonly used as its performances are actually reasonable in finite pool classification [Nguyen et al., 2015a].

Applying them to our problem, we proceed in a single run where we ask each worker to vote on all criteria C for a set of papers. Each worker provides at most N_l labels, and we collect J votes per criteria and per paper. Classification proceeds by evaluating each criterion $c \in C$ on each paper i and, based on the responses received, estimating with one of the mentioned algorithms the probability $P(i \in OUT_c)$ that paper i is classified as out by criterion c.

Once we have probabilities for each criterion, we compute the probability $P(i \in OUT)$ that a paper i should be excluded as the probability that at least one criterion applies (we assume criteria application is independent):

$$P(i \in OUT) = 1 - \prod_{c \in C} P(i \in IN_c) \tag{3.2}$$

The loss ratio skews our classification decision to err on the side of inclusion (for $lr > 1$). The expected loss per paper we suffer for an erroneous inclusion of a paper i is $P(i \in OUT)$, while for an erroneous exclusion it is $lr \cdot (1 - P(i \in OUT))$. This means that our threshold for classifying a paper as OUT is when these quantities are the same, that is, $P(i \in OUT) = lr/(lr + 1)$.

Altering the number of votes per worker N_l, votes per item J, and number of tests N_t will modify the expected price and loss. More tests ideally lead to more accurate workers, more labels mean more accurate classification, and more votes per person enable a more accurate estimation of a worker's accuracy. To analyze price vs loss, we simulate the behavior of the model with various values of N_l, J, N_t, and apply EM, TruthFider (TF) or Majority Voting (MV) to classify papers, and compute the estimated loss. Since

values of N_t and J correspond to a cost, we can also get the price tag corresponding to this loss. Out of this set of price/loss points, we can take the pareto-optimal ones and plot them so that authors can decide which one best fits their needs. As discussed there are cost penalties and practical constraints that do not allow us to set these parameters to arbitrarily high values, and values of N_t and J above 10 do not generate significant improvements [Krivosheev et al., 2017], so the number of reasonable alternatives is fairly small. To simulate the data we need either to make assumptions on the crowd accuracy as well as on criteria power and difficulty, possibly based on prior knowledge, or to estimate these parameters [Krivosheev et al., 2017] by crowdsourcing labels for a few papers (fifty papers already enable a good estimate as shown later). Figure 3.1 shows the results of applying the three mentioned algorithms for 3 and 5 labels per item and criterion (the caption describes simulation parameters). The impact of choosing a specific algorithm is relatively small with the exception of MV performing better when labels per paper and per worker are few, which is a known behavior [Jin et al., 2017b]. The dots represent different number of tests (from 2 to 10) and the arrows shows the direction of growth, from top-left to bottom-right. Some points are Pareto-optimal, so in an interaction with SLR authors we would only show those points and ask for the preferred loss/price point.

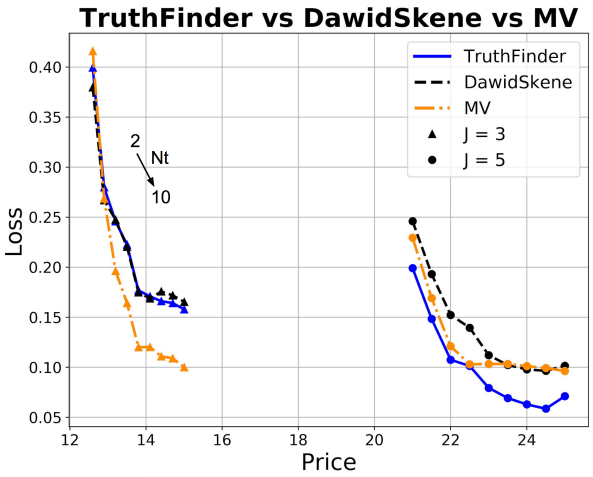


Figure 3.1: Performance of classification algorithms. Simulation with 1000 papers, four criteria of power $= [c1 = 0.14, c2 = 0.14, c3 = 0.28, c4 = 0.42]$, $Nt = [2, 3, .., 10]$, $lr = 5$. Workers are assumed to be cheaters with probability 0.3, and the rest has uniform accuracy in (0.5-1). Accuracy on OUT papers are 10% higher, as seen in experiments.

The results vary slightly if the parameters of the problem and algorithms are different (such as different power, difficulty distributions across criteria, proportion of papers to be excluded, number of papers per worker). We discuss how quality and cost vary later in the paper when we compare and discuss algorithms.

Algorithm 1: **M-Runs Algorithm**

Input: Items I, Criteria C, loss ratio lr
Output: Classified items CI

(1) $CI \leftarrow \{\}$, $UI \leftarrow I$, $thr = \frac{lr}{lr+1}$, I^0 =100 randomly selected papers from I
(2) ***# Baseline iteration (Run 0)***
(3) $V^0 \leftarrow$ collect J votes on I^0 for all criteria C
(4) $CI^0 \leftarrow classify\_items(V^0, thr)$
(5) $CI \leftarrow CI \cup CI^0$, $UI \leftarrow UI - CI^0$
(6) **foreach** $c \in C$
(7) $\hat{\theta}_c \leftarrow \frac{\sum_{i \in I^0} P(i \in OUT_C)}{|I^0|}$, $\hat{\alpha}_c = \frac{\sum_{w \in W_c^0} \alpha_{w,c}}{|W_c^0|}$ (W_c^0 is the set of workers who passed test questions and provided at least one label in baseline iteration)
(8) **#Ranking** $criteria\_order \leftarrow estimate\_best\_order(\hat{\theta}, \hat{\alpha})$
(9) ***#M-Runs iterations***
(10) **foreach** $c \in criteria\_order$
(11) $V^c \leftarrow$ collect J votes on UI on c
(12) $CI^c_{out} \leftarrow exclude\_items(V^0, thr)$
(13) $CI \leftarrow CI \cup CI^c_{out}$, $UI \leftarrow UI - CI^c_{out}$
(14) $CI_{in} \leftarrow$ tag UI as *"IN items"*
(15) $CI \leftarrow CI \cup CI_{in}$
(16) **return** CI

3.3.2 Multi-Run Strategy by Criteria

The multi-run strategy follows the footsteps of the above-mentioned approaches for query optimization in crowd databases that identify the most selective criteria and query based on those first. The difference here is that we also estimate and consider accuracy (we do not want to query the crowd if this brings high disagreement, as it is less cost-effective), and that we work with a specified loss function and a price vs loss trade-offs that are based on the authors' choice. The algorithms proceeds as follows.

Baseline iteration. We first estimate power and difficulty via a *baseline iteration* (run $k = 0$) on a randomly selected subset I^0 of the set of candidate papers I, as shown in Algorithm 1 (We will get back later in the chapter about identifying how large should I^0 be).

In step 4 we classify items and estimate accuracy of each worker with a classification algorithm that also provides accuracy estimates such as TruthFinder (TF) [Dong et al., 2013]. As TF estimates class probabilities, we estimate the power of a criterion c as the expected value of the probability that the criterion applies, as shown in step 7.

We refer instead to the difficulty of a criteria c through the average workers' accuracy on the given criteria, i. e., the average probability that a user, who passed the test questions, gives correct votes on that criteria.

Criteria ranking. Finding the best ordering is trivial if one criterion is more powerful *and* easier than another. Otherwise, different ordering may lead to price/loss points that are on the Pareto frontier and need to be shown to authors for decision. The number of criteria is often low so that considering permutations of all cases where the ordering is not trivial is tractable. We do so in step 8, by computing for each ordering the expected price and loss for different values of N_t and J. The computation of price and loss can be done as for the previous algorithm. Notice that the ordering is very important: given an ordering of criteria $OC = c_0, c_1, ..c_n$, the probability of erroneously excluding an item (probability of false exclusion, or PFE) is the probability of erroneously excluding it in the first round (on c_0), plus the probability of correctly including it after c_0 but erroneously excluding it after c_1, and so on. More formally, denoting with PFE_c the probability of erroneous exclusion when processing criteria c and with PIN_c the probability of classifying a paper as IN on criteria c:

$$PFE = PFE_0 + \sum_{m \in 1,2..n} PFE_m \prod_{j=0}^{m-1} PIN_j \tag{3.3}$$

PFE therefore decreases with PIN, and in practice it decreases sharply if we screen high power criteria first, given that criteria powers over 30% are quite common.

Crowdsourcing iteration. The algorithm iterates through the criteria, excluding items (classified again based on TF in step 12).

The results of M-runs (orange) compared with the baseline single run algorithm (blue) are shown in Figure 3.2b and 3.2c, showing loss and precision vs price for different values of N_t and J. The simulation parameters are the same as previously described. The savings are of approximately 20%, and are in general higher if the criteria diversity in terms of power and accuracy is higher.

3.3.3 Short Adaptive Multi-Run

The previous algorithms apply the same strategy to all papers left to classify. The Short Adaptive Multi-Run algorithm (SM for short)(Algorithm 2) defines instead an *individual* strategy for each item to be labeled, aimed at identifying the shortest path to decision.

The idea is that as we collect votes we understand more about the statistical properties of the overall SLR task (such as criteria power and difficulty) and also of each specific paper, based on the votes obtained for that paper so far. Therefore, we can estimate which is the criterion to test next for each paper by maximizing the probability of (correctly) classifying it as out in the next run, and we can even decide to give up on a paper (leaving it in) because we realize it is too hard (or too expensive) for the crowd to reach consensus or because the probability that we will classify it as out are low. In other words, we aim at excluding the papers for which we can do so cheaply and confidently, and leave the rest to the experts (authors).

At an extreme we would like each run to be composed of one vote on one paper for one criterion (hence the name "short run"). Every time we get a vote we learn something new, and we can use this knowledge to optimize the next vote we ask. In practice a run cannot ask for one vote if we use the basic setup of typical crowdsourcing engines (it would not make sense to take time out of a person to explain a task and a criterion and stop after one vote)[3].

In the following we introduce SM (see Algorithm 2) by first presenting the intuition behind each step and then showing the related math.

We begin at iteration 0 with an empty set of classified items, both in and out: $CI^0_{in} \cup CI^0_{out} = \emptyset$. We assume that authors set thresholds for false inclusion and exclusions, that is, values $\overline{P_{out}}$ and $\overline{P_{in}}$ so that we classify a paper i as out if $P(i \in OUT) \geq \overline{P_{out}}$, and analogously for $P(i \in IN)$.

Baseline estimation. We perform a small baseline run as in the previous approach, to estimate power $\hat{\theta}^0_c$ and difficulty (accuracy) $\hat{\alpha}^0_c$ for each criterion (Algorithm 2, step 2). Experiments have shown us that a baseline of 50 items is often sufficient as an initial estimate (as discussed in the following section), considering also that we revise the estimates as we proceed.

Exclusion probability estimation. Here we begin the iterations. Before each run of crowdsourcing we try to identify, for each item, and given the votes V_i obtained so far for each paper i, which criterion is more likely to *efficiently* filter a paper. In other words, we identify for each criterion c the minimal number $N^{min}_{i,c}$ of successive out votes we need so that *if* we add $N^{min}_{i,c}$ to V_i (resulting in a "imaginary" set of votes V'_i) *then* $P(i \in OUT|V'_i) > \overline{P_{out}}$, and therefore we exclude the paper and stop working on it. Intuitively, for each item we want to select criteria that have a low $N^{min}_{i,c}$ (low number of votes and therefore low cost) and a high probability $P(N^{min}_{i,c})$ of getting those out votes.

[3]With ad hoc implementations, either stand-alone or on top of commercial engines, and with fast estimation it might be possible to achieve one-vote runs though the key optimization here lies in the personalized strategy: the most important aspect is not so much asking at most one vote in each run, but asking one vote per paper

Algorithm 2: **SM-Runs Algorithm**

Input: I, C,lr, $\overline{P_{out}}$, $\overline{P_{in}}$
Output: $CI = \{CI_{in}, CI_{out}\}$

(1) $CI \leftarrow \{\}$, $UI \leftarrow I$, $k \leftarrow 0$
(2) ***# Baseline iteration (Same as Algorithm 1 baseline)***
(3) $\rightarrow CI, V^0, \hat{\theta^0}, \hat{\alpha^0}$
(4) **foreach** $i \in UI$: $P^0(i \in IN_c/V^0_{i,c}) \leftarrow (1 - \hat{\theta}^0_c)$
(5) ***#SM-Runs iterations***
(6) **while** $UI \neq \varnothing$
(7) $k \leftarrow k + 1$
(8) **foreach** $i \in UI$
(9) $c(i) \leftarrow \arg\max_{c \in C} \frac{P(V^{k+1,k+n}_{i,c} = OUT)}{N^{min}_{i,c}}$
(10) check_stop_condition_on_i
(11) $I^k \leftarrow$ N items with highest p(i)
(12) **foreach** $i \in I^k$
(13) $v^k_{i,c} \leftarrow$ collect a vote for c on i
(14) $V^k_i \leftarrow V^{k-1}_i \cup v^k_{i,c}$
(15) $P^k(i \in IN/V^k_i) \leftarrow \prod_{c \in C} P(i \in IN_c/V^k_{i,c})$
(16) $P^k(i \in OUT|V^k_i) \leftarrow 1 - P(i \in IN/V^k_i)$
(17) **if** $P(i \in IN|V^k_i) > \overline{P_{in}}$
(18) $CI_{in} \leftarrow CI_{in} \cup \{i\}$
(19) $UI \leftarrow UI - \{i\}$
(20) **if** $P(i \in OUT|V^k_i) > \overline{P_{out}}$
(21) $CI_{out} \leftarrow CI_{out} \cup \{i\}$
(22) $UI \leftarrow UI - \{i\}$
(23) update power as per algorithm 1, step 7
(24) $CI^{diff\_items} \leftarrow$ tag UI as *"IN items"*
(25) $CI \leftarrow CI \cup CI^{diff\_items}$
(26) **return** CI

Notice that every vote on (paper i, criterion c) we get will move $P(i \in OUT)$ closer or further away from the threshold $\overline{P_{out}}$. This will change our N^{min} and possibly the selected criterion for the next round. The probability of getting an out vote for (i, c) also changes, and it does so more strongly when the accuracy for that criterion is higher.

More formally we proceed as follows. If we denote with k the number of iterations run thus far, and with $V_{i,c}^k$ the votes obtained in the first k runs, then by applying Bayesian rule we have:

$$P^k(i \in IN_c|V_{i,c}^k) = \frac{P^k(V_{i,c}^k|i \in IN_c) * (1 - \hat{\theta}_c^{k-1})}{P^k(V_{i,c}^k)} \tag{3.4}$$

In the formula, after the first run (k=1), the term $\hat{\theta}_c^{k-1}$ is the proportion of papers to which criteria c applies, as computed after the baseline. $\hat{\theta}_c$ is then updated after each run.

The two P^k factors on the right side of Equation 4.3 can be determined as follows, where $J_{i,in}^c$ denotes the number of items i labeled as *in* for criterion c:

$$P^k(V_{i,c}^k|i \in IN_c) = \binom{J_i^c}{J_{i,in}^c} (\overline{\alpha}_c)^{J_{i,in}^c} * (1 - \overline{\alpha}_c)^{J_{i,out}^c} \tag{3.5}$$

and

$$\begin{aligned} P^k(V_{i,c}^k) = P^k(V_{i,c}^k|i \in IN_c) * (1 - \hat{\theta}_c^{k-1}) + \\ P^k(V_{i,c}^k|i \in OUT_c) * \hat{\theta}_c^{k-1} \end{aligned} \tag{3.6}$$

Now that we know how to compute $P^k(i \in IN_c|V_{i,c}^k)$ and therefore $P(i \in OUT|V_{i,c}^k)$ from Equation 4.4, we can compute how the exclusion probability changes as we add n=1,2,.. out votes to $V_{i,c}^k$ obtaining a set we denote as $V_{i,c}^{k \leftarrow n}$ and stop when n is such that $P(i \in OUT|V_{i,c}^{k \leftarrow n}) > \overline{P_{out}}$.

To assess the probability of getting $N_{i,c}^{min}$ out votes for criteria c on item i we proceed by first computing the probability that the next vote is out, as follows[4] (all probabilities are conditional to the votes obtained thus far $V_{i,c}^k$):

$P(v_{i,c}^{k+1} = OUT) = \alpha_c * (1 - P_i^k(I \in IN_c)) + (1 - \alpha_c)P_i^k(I \in IN_c))$.

We then iterate over this formula for getting the probability for the next out votes, remembering that $P_i^k(I \in IN_c)$ will have changed due to the additional out vote.

Ranking. We rank criteria for each item by weighing cost ($N_{i,c}^{min}$) and probability of success (probability $P(V_{i,c}^{k+1,k+n} = OUT)$ of getting $N_{i,c}^{min}$ consecutive out votes). We define the value of applying a criterion as the price we have to pay for unit of probability of classifying the item as out in the next $N_{i,c}^{min}$ votes, that is: $Value_{i,c} = P(V_{i,c}^{k+1,k+n} = OUT)/N_{i,c}^{min}$ We then borrow ideas from predicate ranking optimization in query processing [Hellerstein and Stonebraker, 1993] that essentially ranks based on selectivity/cost (although here we do so per item and assess it at each iteration). Applying the same logic we look for each paper for the criterion with maximum value: $Value_i = \max_{c \in C} P(V_{i,c}^{k+1,k+n} = OUT)/N_{i,c}^{min}$

[4]To simplify the presentation here we take a single value for criteria accuracy as opposed to a confusion matrix.

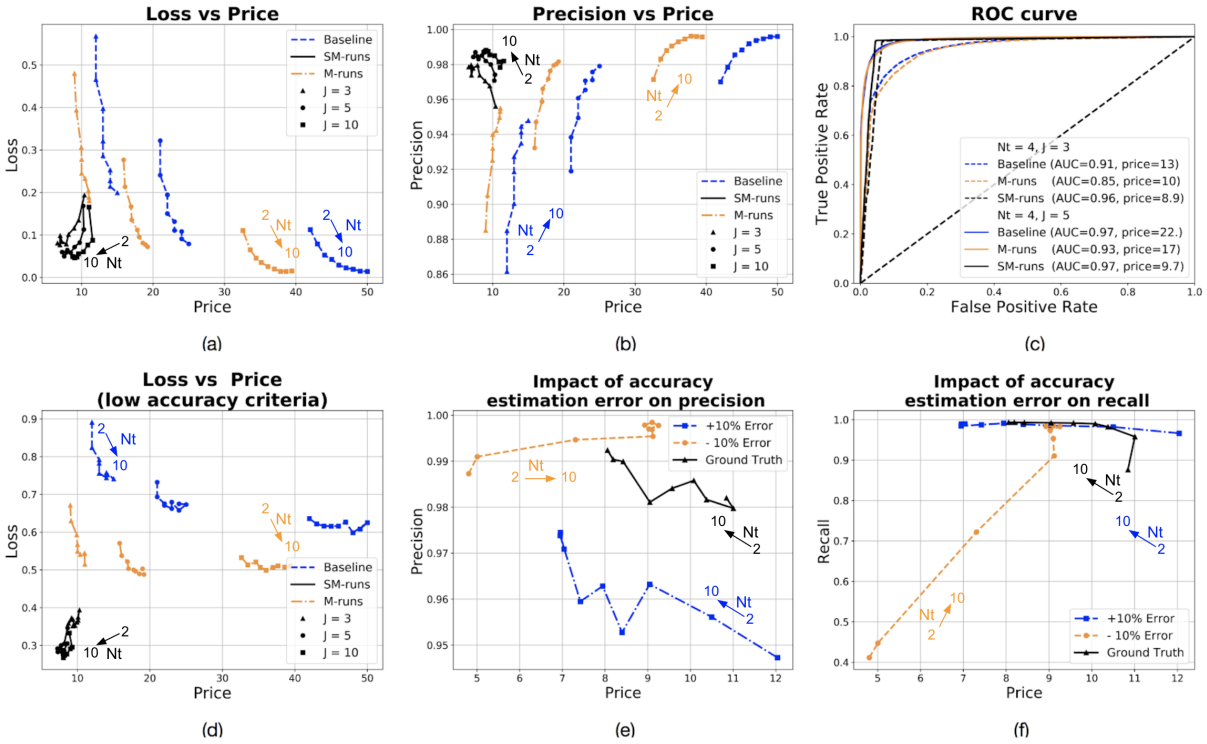


Figure 3.2: Behavior of algorithms. Charts are simulated with 1000 papers, four criteria of power $= [c1 = 0.14, c2 = 0.14, c3 = 0.28, c4 = 0.42]$, $Nt = [2, 3, .., 10]$, $lr = 5$. Workers are assumed to be cheaters with probability 0.3, and the rest has uniform accuracy in (0.5-1). Accuracy on OUT papers are 10% higher, as seen in experiments. See text for description.

In developing SM we explored alternative approaches: a main one we explored involves estimating how $P(i \in OUT)$ is likely to change if we ask for one vote on c, as an attempt to drive our choice for which vote to ask next. With relatively simple math, we can estimate the probability of the next vote being in or out, and the impact that this has on $P(i \in OUT)$, and we can select the criteria that leads us closer to the threshold. This initial choice however has an undesired behavior: if there is a low accuracy, high power criterion, it leads us to choosing this criterion. However, the low accuracy means we only take little steps towards our threshold, making the walk long and expensive. Instead, we choose criteria that can provide large variations towards the out threshold.

Stopping. As we iterate, we can see that $Value_i$ may be so low (for example, if we get conflicting votes) that it becomes ineffective to poll the crowd for that item. We can therefore stop working on papers for which $Value_i$ is lower than a threshold based on authors' preferences (Notice that we disregard the money already spent on a paper, as that is a *sunk cost* [Arkes and Blumer, 1985]). The reasonable threshold here depends on the cost ratio cr of the crowd cost for a single vote on one paper and criterion (PPL from

Formula 3.1) divided by the author classification cost. The lower the cost ratio, the more convenient it is to insist with the crowd. For typical cost ratios, considering classification costs as estimated in the literature (see, e. g., [Mortensen et al., 2016]) of around 2$ per abstract (for the US, in the medical field and including overhead), a good empirically set threshold is 100.

Crowdsourcing iteration. Ranking determines the priority for the next batch of votes. The batch size is the minimal size that can practically be achieved while ensuring each worker gets value for the time they spend learning and doing the task. In practice, it rarely makes sense to offer batches of less than 10 items as they are less attractive. We return to the crowd to ask one more vote for each paper in the batch, determine the probability of exclusion as discussed above and classify paper as out if $P(i \in OUT) > \overline{P_{out}}$. If there are no more papers left to classify we stop, else we iterate.

We next analyze the results of the algorithm and discuss its properties, also in light of crowdsourcing experiments.

3.4 Analysis and Experiments

3.4.1 Simulations

We first show the behaviors of algorithms via simulations. The strategies presented here have a number of parameters and the behavior varies in interesting ways as we change these parameters. We also remind that authors do not set or estimate any parameter: they simply need to state their loss function and the preference for given loss vs price points when there is no Pareto-optimal value.

Figure 3.2 shows the result of a simulation run with 1000 papers and parameters as described in the caption. It plots the loss vs price curve for the SM strategy for the same scenario discussed for the other algorithms (The SM variant adopted here has a 1000 papers run, assumes a stopping threshold of 100, and shows an average of 50 simulations). $\overline{P_{out}}$ is 0.99. Figure 3.2(a) and (b) show that SM can achieve the same loss and precision for a fraction of the cost (both could improve by changing $\overline{P_{out}}$, though increasing the price)[5]. Notice that price and loss both decrease (at least initially) as we increase the number of tests N_t, which is our "knob" to increase accuracy (and cost) of workers. This is because SM detects the increased accuracy and adapts to it by asking for less votes for the same loss and precision. Figure 3.2c shows the ROC curve where we can see that SM has a greater area for a much smaller price. Charts are analogous in terms of shapes and trends for other values of θ, J (and for N_t as well for the ROC curve) so we do not show

[5]We omit plotting the std bars as they would make the chart unreadable

them. Figure 3.2d shows again loss vs price but this time assuming the presence of a very difficult criterion (accuracy of 0.55), and shows the robustness of SM on the loss (even with the very conservative loss ratio of 5 we used here).

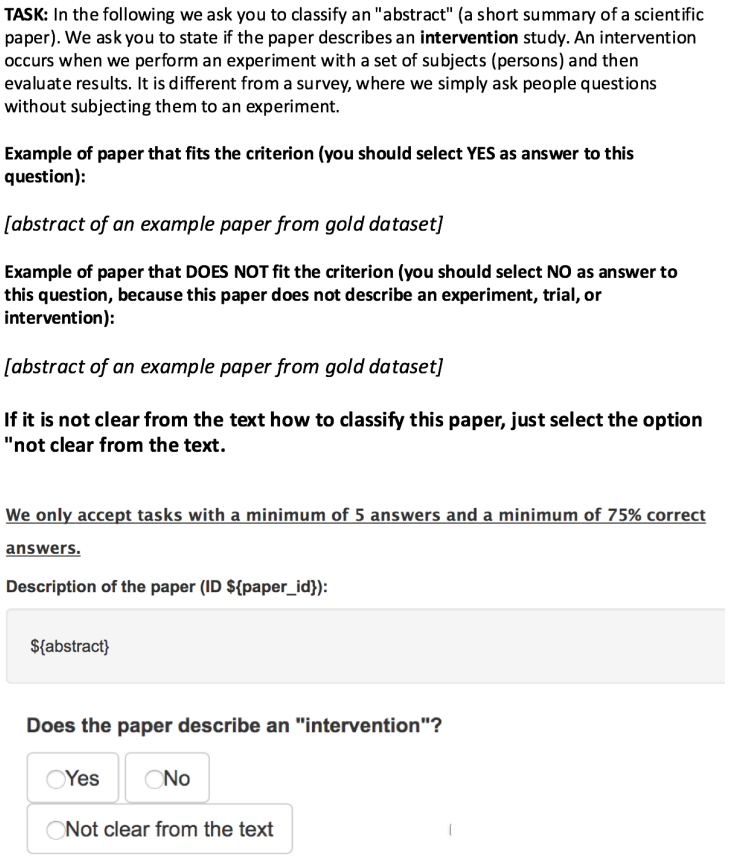


Figure 3.3: Classification task for SLR.

Figure 3.2e and 3.2f show the impact of estimation error for accuracy on precision and recall respectively. Notice that if we underestimate (orange line) we achieve higher precision (we are more conservative). For recall, if we underestimate accuracy and accuracy is low (N_t is low) then we get very low recall: we give up rather early, leaving papers to authors to classify. As accuracy increases, the differences smooth out and are within the variance. The charts for power estimation error have essentially the same shape and are not shown.

Baseline runs and numbers of labels per paper affect the estimation errors. The issue is not so much the number of papers in the baseline: 40-50 papers suffice to estimate power within a 5-7% margin of error (consider the problem similar to estimating the fairness of a coin modeled as a Beta distribution, 50 tosses would give a reasonable estimate). Furthermore, estimates are re-assessed as we go. The key here is rather to enable a good

accuracy estimation, and experiments have shown that with less than 3 votes per paper the estimation error grows above 10% and with low accuracy criteria this can generate very low recall (as we may believe accuracy to be at 0.5). Experiments with variations of the stopping threshold also produced limited effect. When going from 100 to 150, the recall increased by approximately 0.04%, always keeping the precision threshold at 0.99. The price difference is also negligible.

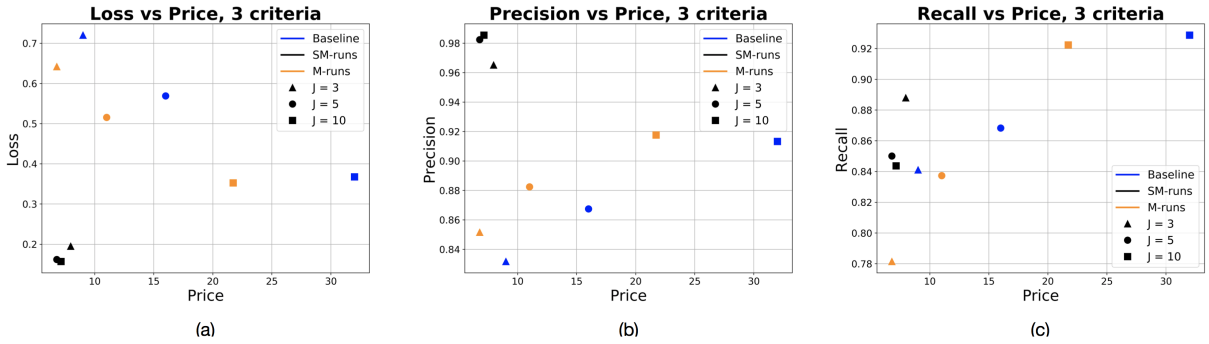


Figure 3.4: Behavior of SM with data from experiments. Runs of 1000 papers, accuracy and power as described in the text, the other parameters are unchanged.

3.4.2 Crwdsourcing Experiments

During 2017 we performed a set of studies and experiments on two commercial crowdsourcing engines (Figure Eight and Mechanical Turk). We ran a total of 20 experiments with different settings, asking workers to label a total of 174 papers with two to four exclusion criteria (a total of 514 classification decisions) taken from two systematic reviews, one done by us in an interdisciplinary area (computer science and social sciences) reviewing technology for fighting loneliness, and the other in medicine [Veronese et al., 2017] having more complex exclusion criteria. We collected votes by over 3200 respondents. These initial studies helped us to understand the nature of the problem, estimate crowd accuracies, get a feeling for latencies and costs, and also refine task design which, although orthogonal to our goals here, is important for getting good results [Yang et al., 2016]. In the following we focus on the experiments to assess the validity of SM with respect to other algorithms and baselines.

Setup. To this end we classified 374 papers on AMT by posting tasks that were asking crowd workers to classify many papers based on one criteria. We requested workers with an HIT approval rate of 70% or more. The task starts by explaining the criteria to workers, providing a positive and a negative example, and then asking to label the papers as in, out, or unclear from the text (Figure 3.3). Adding the latter option was a result of previous

experiments where many workers complained that this option was missing and were unsure about what to answer to qualify for payment. We also informed workers that they would need to get 75% of the answers as correct in order to be paid. The examples are taken from the "gold" dataset, which are the papers classified by researchers in our team. Each worker saw the same example papers in the instructions for the same criteria. The three criteria we tested involved assessing whether the paper included an *intervention* (as shown in Figure 3.3), whether it described studies on adults *65 and older*, and whether it involved the use of *technology*.

The task proceeded by criteria, not by paper: we showed instruction for a criterion, and then asked for classification on that criterion. We chose this option as we noticed in initial experiments that explaining and understanding criteria takes effort, and teaching workers the subtleties of several criteria at the same time may lead to increased effort and reduced accuracy. Workers could classify the papers until they wished to do so. We repeated the process for the three criteria getting at least 5 votes per paper (we collected up to 15 votes per paper for the intervention criterion show in the figure as it had low accuracy and we wanted to analyze it more deeply). Before running the task we did some pre-runs to assess the proper pay. In terms of costs, we experimented different payments, always making sure that we stay well over 8USD per hour based on estimated completion times, which results in approximately 10 cents per vote. We did not screen workers with test questions, though the experiments gave us a dataset over which we can now use to "simulate" the effect of filtering out workers that did not get 100% accuracy on the first N_t questions.

Results. All tasks were completed in a few hours, and we assume that if we had more papers they would have been classified with sublinear increment in time. As expected (see table below), and consistently with the literature [Mortensen et al., 2016], power and workers' accuracy vary significantly by criteria. Use of technology has high power as words related to technology are very common and it is hard for keyword-based queries to filter for the specific use of technology we look for (Table 3.1).

Criteria:	*intervention*	*use of tech*	*65 and older*
Power	0.24	0.61	0.05
Accuracy	0.60	0.77	0.75

Table 3.1: Power and Accuracy of criteria

The presence of a criteria (intervention) with rather low accuracy underscores the importance of an adaptive approach where we focus on high-accuracy criteria and leave the leftover papers to the authors.

We additionally studied a bit the average agreement among expert raters for the screening task of scientific abstracts. Mateen et al report on an experiment that measured agreement on around 96% of the papers [Mateen et al., 2013]. An analysis of SLRs conducted by our team reported 92% agreement among two raters and, in addition, 2% of cases where one rater was unsure. This indicates that the $\overline{P_{out}}$ precision threshold we picked of 0.99 is in line or even exceeding current standards. We also observed that the mere act of having the need to explain criteria to others (that also demand fairness in job acceptance) forced us to be very precise and indeed, looking back at our own classification, we found errors also due to a certain imprecision in the initial definitions.

Using the experimental data to fuel simulations did not bring significant changes to the charts already discussed, although there are interesting differences and we focus on these in the following, especially to underline the limits of SM. One interesting aspect is that actual data do not precisely and consistently fit the model: workers accuracy cannot always be modeled as i.i.d. variables, and the margin of error in predicting future accuracy from past is rather high, even if we vary the testing patterns. This is not entirely surprising as some workers may improve as they proceed with the task while others may get sloppy or tired, and indeed optimal testing to cater for these issues is an active area of research [Bragg et al., 2016].

Figure 3.4 shows the results of such experiments-fueled simulations, assuming no additional tests filtering. Figure 3.4a shows the usual loss vs price chart, and the results are fairly consistent with the simulations. Figures 3.4b and c break this down by precision and recall. The latter is particularly interesting as the recall for baseline is somewhat comparable to M-runs, or at least it makes for a non obvious choice for lower level of J. This is probably due to the relatively low accuracy we have in these (relatively untested) set of workers.

Finally we observe that in terms of overall cost, even at 10 cents per vote we remain well below author cost for the paper we screen out (from 20 to 40%).

3.4.3 Conclusion

In this chapter, we tested the feasibility of crowd workers in performing the filtering stage of Systematic Literature Reviews to identify papers to be included in the review based on a set of exclusion criteria. We proposed and analyzed a set of strategies for crowd-based SLR screening. We validated both applicability and results of the approach through a set of crowdsourcing experiments and discussed properties of the problem and algorithms that we believe to be generally of interest for classification problems where items are classified via a series of successive tests.

The proposed multi-run strategy by criteria exploits differences in criteria power and

difficulty when difficulty and power are a priori unknown. Therefore, we first estimated them, and then select an optimal ordering of the criteria, so to be able to apply them one by one (one run per criterion), each time taking into considerations only the papers not excluded by the previous run.

The shortest-run algorithm defines an individual strategy for each paper to be labeled, aimed at identifying the shortest path to decision. With each new coming vote, we update our estimation of the parameters and identify for each paper the criteria for which a vote takes us closer or past our exclusion threshold. We also can rank papers based on this estimation, and identify the low hanging fruits.

The shortest-run algorithm has the potential to outperform a number of non-adaptive approaches in terms of cost and accuracy for finite pool classification problems, and especially for SLR. We also confirm initial findings that crowdsourcing is feasible for paper screening in SLR. The work still has many limitations, especially that of improving the estimation of accuracy of workers and extending the model to cover the case where workers' accuracy is very "noisy".

Chapter 4

Combining Crowd and Machines for Multi-predicate Item Screening

A frequently occurring classification problem consists in identifying items that pass a set of *screening tests* (filters). This is not only common in medical diagnosis but in many other fields as well, from database querying - where we filter tuples based on predicates [Parameswaran et al., 2014], to hotel search - where we filter places based on features of interest [Lan et al., 2017], to systematic literature reviews (SLR) - where we screen candidate papers based on a set of exclusion criteria to assess whether they are in scope for the review [Wallace et al., 2017a]. The goal of this chapter, is to understand how, given a set of trained classifiers whose accuracy may or may not be known for the problem at hand (for a specific query predicate, hotel feature, or paper topic), we can combine machine learning (ML) and human (H) classifiers to create a hybrid classifier that screens items efficiently in terms of cost of querying the crowd, while ensuring an accuracy that is acceptable for the given problem. We focus specifically on the common scenario of *finite pool* problems, where the set of items to screen is limited and where therefore it may not be cost-effective to collect sufficient data to train accurate classifiers for each specific case. In this chapter, we will often take the example of SLRs mentioned above, which is rather challenging in that each SLR is different and each filtering predicate (called *exclusion criterion* in that context) could be unique to each SLR (e. g., "exclude papers that do not study adults 85+ years old").

The area of crowd-only and of hybrid (ML+H) classification has received a lot of attention in the literature. Research in crowdsourcing has identified how to best classify items given crowd votes [Liu and Wang, 2012; Dong et al., 2013; Whitehill et al., 2009], including work that focuses specifically on the *screening* problem discussed here [Parameswaran et al., 2014; Wallace et al., 2017a; Mortensen et al., 2016; Krivosheev

et al., 2017; Lan et al., 2017]. Interesting ways to combine ML and H classifiers have also been proposed, for example by building ML classifiers that can also include crowd votes as features, whose purpose is not only that of providing better classification but also of weighing the value (vs the cost) of obtaining additional crowd votes [Kamar et al., 2012]. Other hybrid approaches ask the crowd to extract interesting patterns to be then fed to algorithms for classification, as opposed to relying on ML to do this [Cheng and Bernstein, 2015; Rodriguez et al., 2014].

The approach we propose leverages the information provided by each kind of classifier (machine and human) to improve the effectiveness of the other kind so that they can be stronger together. The algorithm is based on a probabilistic model that adapts to the specific characteristics of each item, screening filter, workers' accuracy, and ML classifier available to identify *what* to ask next for each item (which filter to test) and *if* we should stop polling the crowd for a given item, either because we reached a decision or because we realize we cannot do so cheaply and confidently (that is, the problem of classifying that item is too hard for the crowd-machine ensemble). The latter point is particularly important: there may be items, filters, or classification problems in general where it is questionable whether machines and/or the crowd can classify with acceptable accuracy. If the algorithm can identify this subset of items and filters - or (item, filter) pairs - early and avoid spending money on them, then we know we can confidently try a crowd or hybrid approach as it will not waste money.

In essence, the approach - which we call *hybrid shortest run* (HSR) - works by considering the output of machine classifiers as a prior to the class probability for each item. This information is combined with an adaptive crowdsourcing algorithm that refines the class probability by polling the crowd on the (item, filter) pair that makes it more likely to reach a decision cheaply. In turn, classification information that is progressively obtained as we poll the crowd can be used to re-assess the performance of ML classifiers as well as create a model over the available classifiers for improved accuracy.

While many adaptive algorithms have been proposed (e. g., [Kamar et al., 2013; Dai et al., 2013; Nushi et al., 2015]), to the best of our knowledge this work is the first to discuss hybrid classification in screening contexts, which has unique requirements and opportunities since, as we will see, the heart of the problem lies in identifying the filters (e. g., the exclusion criteria in SLR) that i) are most selective (screen out a large proportion of papers) and ii) that crowd or machines can classify accurately (can correctly determine if the exclusion criterion applies to a paper). If we can do so, we can focus on getting crowd votes for these filters first, leaving a smaller set of items for the filters the crowd finds hard to classify correctly or that do not have high selectivity.

We believe this work is also the first to take a "black box" approach to hybrid

classification: since we tackle finite pool problems, we do not assume we can develop an accurate, dedicated classifiers for each classification problem, although we may have available (possibly weak) classifiers from similar problems - in our SLR example these may be neural nets developed for other SLRs in the same or different fields or for other exclusion criteria. Rather, we identify how to leverage classifiers we are given and for which the accuracy for our task is unknown. We do this by a combination of (minimal) testing to filter out very poor classifiers and by combining the remaining ones to set a prior probability for whether a filter applies to an item. This helps us determine the most promising (item, filter) pairs for which asking a vote to the crowd has the highest probability of reaching a decision cheaply and confidently.

While it is not surprising that hybrid approaches have the potential to outperform human- or machine-only methods, the contribution we provide here lies in showing how we can cope with finite pools problems where potentially weak classifiers - and even classifiers with unknown accuracy for the problem at hand - can be leveraged effectively, even in contexts characterized by very demanding requirements in terms of accuracy (as is the case for SLRs) and in the presence of difficult problems where also the crowd has low accuracy.

HSR always estimates the characteristics of the problem and the classifiers, and identifies which are the items and filters for which it can draw accurate conclusions cheaply, leaving the rest to expert classification.

We also study how the effect of incorporating classifiers (and ensembles of classifiers) into a crowdsourcing algorithm varies based on the different characteristics of the problem at hand (such as different selectivity of filters, accuracy of classifiers, correlation among base classifiers, amount of data available for testing classifiers and for training ensembles) and show why and under which conditions *ensembles* of classifiers do or do not provide benefits over "simply" using the best available classifier at hand.

4.1 Background

4.1.1 Crowd Classification in multi-predicate problems

The set of classification problems we address here is that of *finite pool* classification [Nguyen et al., 2015a]: we work on a finite set of items, and the classification criteria may be unique to the problem. SLRs are one case of such problems that has been thoroughly investigated by many researchers in recent years [Mortensen et al., 2016; Krivosheev et al., 2017; Nguyen et al., 2015b; Sun et al., 2016].

We borrow several concepts from this work and indeed we build over a slightly modified version of their algorithm. However, crowdsourced classification algorithms for us are to a large extent swappable components we leverage: we do not aim at improving crowd-only

classification but rather to study how to efficiently combine ML and human classifiers, and specifically how to do so in a manner that is robust to weak classifiers to avoid compromising on accuracy and avoid spending money in classifications not likely to produce good results.

Multi-filter classification has been also widely studied in search and information retrieval contexts. Indeed, an instance of such problem is that of selecting which predicates to apply first when filtering tuples in a query. The seminal work in this area is by Hellerstein and Stonebraker, discussing the problem of ranking predicates in query plans [Hellerstein and Stonebraker, 1993].

More recently, [Franklin et al., 2011], [Park and Widom, 2013] and [Parameswaran et al., 2012, 2014] focused on a similar problem but in the context of crowd-powered databases, while [Avnur and Hellerstein, 2000], [Babu et al., 2004], [Lan et al., 2017] presented *adaptive* extensions that tune the algorithm as the query progresses (see [Deshpande et al., 2007] for a review). We differ in many ways with respect to the prior art in this area. First, again, we aim at combining ML classifiers with crowd. Furthermore, we follow an approach that makes initial estimates on the characteristics of the classification problem and then continuously revise the estimates, both in general and for each item to be classified, to identify an item and filter specific strategy to reduce errors and cost. Importantly, we identify the items on which we should give up trying, since it is more efficient to leave them for experts to screen.

Other prior work also addresses the issue of optimizations in terms of costs for obtaining labels and techniques to reduce cheating [Smyth et al., 1995; Karger et al., 2011a; Hirth et al., 2013; Eickhoff and de Vries, 2013; Hirth et al., 2011]. For example, Hirth and colleagues [Hirth et al., 2013] recommend specific cheating detection and task validation algorithms based on the cost structure of the task. In this chapter, we do not optimize cheating detection or even task design to a large extent, in that those are orthogonal aspects with respect to the methods and optimizations discussed here.

4.1.2 Hybrid Crowd-Machine Classification

Just like crowd classifiers, ML classifiers are also ingredients of a solution we borrow from the state of the art. In the context of our problem the interesting issues are how to combine ML and humans, how to ensemble available classifiers, and how to balance the request cost of (crowd or expert) votes with the need of training and testing ML models.

Hybrid classification is an increasingly active area of research (see [Vaughan, 2017] for a recent survey). Many researchers study the problem in the context of clustering or entity resolution ([Wang et al., 2012; Vesdapunt et al., 2014; Vinayak and Hassibi, 2016; Gomes et al., 2011]). In general, many of these techniques operate by first classifying items

automatically when ML classifiers are highly confident, and by then leveraging ML to shape the kind of task proposed to the crowd (e. g., identifying potential clusters or matching items to propose to the crowd [Vesdapunt et al., 2014; Vinayak and Hassibi, 2016]). Another class of hybrid approaches, also extremely popular in industrial applications, are those where ML makes a proposal or a pre-filtering and humans confirm or refine. This happens in many fields, recently even in fashion [1].

[Kamar et al., 2012] propose instead a promising approach where crowd features (votes, as well as potentially other aspects of the crowdsourcing process such as task completion times) and task features are combined into a broader set of features to be used to learn a model. Researchers also explored using the crowd to extract features and patterns to then be leveraged by classifiers [Cheng and Bernstein, 2015; Rodriguez et al., 2014].

While very interesting, these results are complementary to the work discussed here as we do not aim at developing a good base classifier but at leveraging effectively a set of arbitrarily weak classifiers we are given for the problem at hand. Perhaps the most closely related prior art is the work by [Nguyen et al., 2015a], who adopt a clever mix of crowd+expert+machine learning with an active learning classifier, where papers to be labeled are iteratively chosen to minimize overall cost and loss, by comparing estimated loss of crowd classification versus expert classification. Our problem and approach differs, however, in that we focus on exploiting existing classifiers and in doing so for optimally selecting the (item, filter) pair to poll the crowd on. It is indeed in the optimization of which question to ask the crowd that lies the essence of the approach we propose.

4.2 Model, Assumptions and Problem Statement

We next define the problem we aim at solving and the model, which captures a few broad assumptions we make.

Specifically, we assume to have in input a set (I, F, C, L) where I is the set of items to classify (in our SLR example, these are papers to screen), F denotes the filters (paper exclusion criteria), C represents the set of ML or H classifiers, and $L = k * FE + FI$ is a loss function, modeled as a linear combination of false exclusions FE and false inclusions FI, which may carry a different relative weight k. The reason for the weight k is that in many contexts the two kinds of errors have very different implications. This is for example the case when screening is performed to reduce the number of cases to be brought to human attention, such as potential credit card fraud, tweets/posts potentially linked to criminal activities, or SLRs: screening out an item is often more "costly" than erroneously bringing the item to human attention.

[1] https://multithreaded.stitchfix.com/blog/2016/03/29/hcomp1/

The (possibly empty) set of given ML classifiers is assumed to be trained to receive an (item, filter) pair and output whether the filter applies to the item (e. g., receive a paper and an exclusion criterion and assess if the criterion applies to the paper). As mentioned, we do not discuss how to obtain classifiers (some examples are provided in the experiments section), as this is the subject of ample literature and it anyways depends on the problem domain. We do *not* necessarily assume that the classifiers are trained or even tested in the specific problem at hand (that is, for the specific filters and items under consideration), and therefore in general we do not even know their accuracy.

For human classifiers, we assume we have a set of generic workers of different and initially unknown accuracy and a set of high-accuracy but expensive experts.

Each classifier $c = \{cost, a(f), \rho(f, C)\} \in C$ is associated with the cost of asking one vote on an (item, filter) pair, with a filter-specific estimated accuracy (a 2x2 confusion matrix capturing probability of correct decisions for positive and negative labels), and with its *error correlation* ρ with other classifiers. The error correlation among two classifiers is the probability that both make an error given that one of them makes an error.

Because we have no knowledge on the classifiers' accuracy, we conservatively model it as a uniform $Beta(1, 1)$ distribution for both positively and negatively labeled items (which means we do not even assume classifiers are better than random). If we do have additional information, this can be incorporated in the $Beta$.

Consistently with crowdsourcing literature, we also assume that crowd workers opinions are independent (that is, they make independent errors, given an item, filter and true label).

Each filter f is characterized by *difficulty* d_f and *power* θ_f. Difficulty reflects how easy it is for crowd workers to classify items correctly on that filter. Following Whitehill [Whitehill et al., 2009], we model difficulty as a real number $d_f \in [0, +\infty)$ that, given an expected accuracy α_w of a worker w, skews the accuracy as follows:

$$\alpha_{f,w} = 0.5 + (\alpha_w - 0.5) \cdot e^{-d_f} \tag{4.1}$$

As the difficulty d_f grows, $\alpha_{f,w}$ goes to 0.5, corresponding to random selection, which we consider to be the lowest accuracy level. The power $\theta_f$is simply the proportion of items to which filter f applies. Notice, in this chapter, we do not assume that power affects difficulty and we also do not assume different accuracies based on the true label of the items. Both difficulty and power are unknown and we assume no prior knowledge on them.

Given this model and assumptions, our goal is to identify an algorithm that can efficiently (in terms of cost) query the classifiers available and aggregate results while achieving the quality goals, stated in terms of loss as well as of the classical precision and recall measures.

4.3 Baseline Algorithms

We next describe baseline crowd and machine classification algorithms on which we base the construction of the hybrid approach. The crowd-based one is *shortest run* (SR) [Krivosheev et al., 2018b], a recently developed algorithm that has been shown to perform well for multi-filter screening. We summarize SR here to make the chapter self-contained. There are other reasons why SR is particularly suited for the hybrid approach we propose, and we get back to this in the following.

4.3.1 Recap of Shortest Run

The Short Adaptive Multi-Run algorithm (Shortest run, SR for short) identifies and continuously updates an individual strategy for each item to be screened by identifying the shortest path to decision. In a nutshell, the idea borrows concepts from Partially Observable Markov Decision Processes (similarly to [Kamar et al., 2013; Dai et al., 2013]), by assessing based on the current state (that is, the knowledge accumulated thus far at the *global* level - power and difficulty for each filter - and at the *local* level - the votes on each specific item) which are the (item, filter) pairs to submit to the crowd for testing because they would be more likely to quickly (cheaply) lead to an accurate decision. SR may also decide to quit trying the crowd classification approach on an item if it believes that it would be too expensive to reach an accurate decision.

SR proceeds by performing what authors call a *baseline run* where a small set of items (usually 50) is screened with the standard approach of asking for labels on all filters (typically, 5 labels per item and filter) and classifying using a variant of Expectation Maximization. The result of the run is information on the power θ_f and difficulty d_f (and, correspondingly, workers' accuracy α_f) for each filter.

This information is used to obtain a prior estimate on the probability that the *next* vote for each (item i , filter f) will be a vote to screen out the item (that is, a vote that the filter applies to the item). We get an out vote if the item is out (with probability θ_f) and the worker answers correctly, or if the item is in (with probability $1 - \theta_f$ and the worker answers incorrectly:

$$P(v_{i,f} = OUT) = \alpha_f \cdot \theta_f + (1 - \alpha_f)(1 - \theta_f) \tag{4.2}$$

In addition, by applying Bayes, we know the probability of a filter f screening (i. e., $P(i \in OUT_f)$) or not screening ($P(i \in IN_f)$) an item i, once we obtain a label for the (i, f) pair.

$$P(i \in IN_f | v_{i,f}) = \frac{P(v_{i,f} | i \in IN_f) * (1 - \theta_f)}{P(v_{i,f})} \tag{4.3}$$

Because an item is screened out if at least one filter applies, then:

$$P(i \in OUT) = 1 - \prod_{f \in F} P(i \in IN_f) \tag{4.4}$$

An item is classified as out if $P(i \in OUT)$ is greater than a threshold $\overline{P_{out}}$. These formulas allow SR to estimate both what the next vote can be for a pair (i, f) and the impact each vote has on $P(i \in OUT)$. These estimates are updated as votes come in, where in Formula 4.3 the class probability, initially derived only from filter power and therefore equal for all unclassified items, is updated based on the votes obtained thus far for that item.

The above formulas can be easily extended to compute the probability that the next n votes for an (i, f) pair will be *in* or *out* votes. SR can in particular estimate the minimal number of votes $N_{i,f}^{min}$ in one direction (in or out) it needs to reach a decision, and the probability $P_{i,f}^{min}$ of getting such votes. As $N_{i,f}^{min}$ grows and its probability shrinks, SR may decide that crowd classification cannot be done efficiently, and quits trying (the stopping criteria are based on threshold which can be set as discussed in [Krivosheev et al., 2018b]). Intuitively, items that are left unclassified are essentially not filtered out, so they contribute to a higher loss, which has to be weighted against the price one would incur by insisting with the crowd. This in general is part of the same trade-off of how much users are willing to spend for unit of loss.

4.4 Hybrid SR algorithm

While there is ample literature on how to generate a "good" ensemble of classifiers [Rokach, 2010; Dietterich, 2000], the prior art on the general problem of combining an existing set of ML classifiers $C = \{c\}$ is relatively less abundant [Džeroski and Ženko, 2004]. We base the hybrid machine-crowd classification strategy on modifying SR. In the following, we first motivate why we start from SR and then introduce hybrid shortest run (HSR) by presenting, at each step, first the intuition and then the formalization.

The reason for starting from SR is that i) it was designed for multi-filter screening and has shown to perform better than baseline algorithms for crowd multi-predicate classification, ii) it has per item per filter probabilistic model that can leverage prior knowledge on items and filters, and ML classifiers can provide such knowledge, and iii) the algorithm can adapt to work with different test items T of different sizes. This is

Algorithm 3: **Hybrid SR Algorithm**

Input: I, C, F, T, $\overline{P_{out}}, \overline{P_{in}}$, *Budget*
Output:$CI = \{CI_{in}, CI_{out}\}$ *#classified items*

(1) $CLF \leftarrow$ *Select classifiers based on T tests*
(2) $C_{meta} \leftarrow$ *Ensemble CLF*
(3) $\boldsymbol{Prior}, \hat{\theta^0} \leftarrow$ *run ensemble on I*
(4) $CI \leftarrow \{\}$, $UI \leftarrow I$,$k \leftarrow 0$
(5) $Votes^0, \hat{\alpha^0} \leftarrow$ ***Baseline run***
(6) *#SRuns iterations*
(7) **while** $UI \neq \varnothing$ **and** $Budget \neq \varnothing$
(8) $\quad k \leftarrow k + 1$
(9) $\quad$ **foreach** $i \in UI$
(10) $\quad\quad f(i) \leftarrow$ *assign filter on i —* ***Prior****, Votes*
(11) $\quad\quad$ *do check stop condition on i —* ***Prior****, Votes*
(12) $\quad I^k \leftarrow$ *items with highest prob of classification*
(13) $\quad$ **foreach** $i \in I^k$
(14) $\quad\quad v^k_{i,f} \leftarrow$ *collect a crowd vote on* $f(i)$
(15) $\quad\quad Votes \leftarrow Votes \cup \{v^k_{i,f}\}$
(16) $\quad\quad$ **if** $P(i \in IN | Votes^k_i, \boldsymbol{Prior}) > \overline{P_{in}}$
(17) $\quad\quad\quad CI_{in} \leftarrow CI_{in} \cup \{i\}$
(18) $\quad\quad\quad UI \leftarrow UI - \{i\}$
(19) $\quad\quad$ **if** $P(i \in OUT | Votes^k_i, \boldsymbol{Prior}) > \overline{P_{out}}$
(20) $\quad\quad\quad CI_{out} \leftarrow CI_{out} \cup \{i\}$
(21) $\quad\quad\quad UI \leftarrow UI - \{i\}$
(22) $\quad \hat{\theta^k} \leftarrow$ update power
(23) $CI^{difficult\_items} \leftarrow$ label UI as *"IN items"*
(24) **return** $CI, CI^{difficult\_items}$

important, as test items can help us filter out ML classifiers with an expected accuracy lower than a threshold $\overline{a}$ (to be tuned as discussed later), and the more extensive set of crowd-classified items from the baseline can be used to a) assess independence among ML classifiers (which is necessary if we want to pool votes from ML classifiers with simple algorithms such as majority voting or Naive Bayes), and b) build an ensemble model out of the ML classifiers where the output of each ML classifier is a feature. This is similar to

stacking [Džeroski and Ženko, 2004] although we do not apply it over the training data used to build the individual classifiers (which we do not assume to have), but rather on the labels as estimated by the hybrid classification.

The Hybrid SR pseudo code is presented by Algorithm 3. The first step consists in filtering out classifiers that do not perform well for the case at hand. Specifically, we want to retain a classifier if we are confident it has an accuracy that is better than random. In principle, every contribution that is better than random helps, especially if it is independent from the other classifiers and if we weigh the vote by the classifier accuracy.

As commonly done in crowdsourcing, we assume we have access to a high accuracy test dataset T (obtained at a cost $C_T = ec * |T|$, that increases linearly by a factor ec with the number of test examples, where ec represents the cost for an ideal, expert screening). We then assign a prior probability distribution to each ML classifier and for each filter (ML classifiers may perform differently for different filters), reflecting our belief of how well the classifier performs on the problem at hand. As mentioned, in absence of additional information, we assume a $Beta(1, 1)$ uniform prior for a given filter. Other choices of prior are possible, assuming more likely that classifiers will have an accuracy above 0.5 and that accuracies close to 0 or 1 are very unlikely (for example, a Beta (3, 2) has such characteristics). If we have a non negligible number of test cases the impact of this choice on the performance of the algorithm is relatively small. In the following for simplicity we assume a Beta (1, 1) prior for all filters.

ML classifiers are tested with the gold dataset T (line 1), resulting in posterior probability $Beta(1+correct\_answers, 1+failed\_answers)$ which has a known distribution. We can, in particular, retain classifiers whose probability of being better than random is greater than a selection threshold sc, and this can be easily computed from the Beta distribution. Then we build an ensemble of ML classifiers and run it on a whole unclassified pool of items UI (lines 2, 3).

Consistently with SR, we then perform a baseline run on B items (on all filters), both to estimate crowd accuracy on each filters and to get data for the next step (line 5).

This information is used to inform a prior $p^{mc}_{i,f}$ for each item and filter in Equation 4.3. In other words, while SR takes as prior for each filter f the proportion $1 - \theta_f$ of items classified as "in" for f and computes an overall prior, here we use the baseline ensemble output for each prior $p^{mc}_{i,f}$, where the class probability is the confidence that the filter applies. Equation 4.3 therefore becomes

$$P(i \in IN_f | v_{i,f}) = \frac{P(v_{i,f} | i \in IN_f) * (1 - p^{mc}_{i,f})}{P(v_{i,f})} \tag{4.5}$$

Notice that the prior has three effects: 1) it concurs to determine the classification

probability; 2) it also affects which (item, filter) pairs we pick next for obtaining the crowd vote, since Equation 4.3 also affects $N_{i,f}^{min}$; 3) it has an impact on the stop condition (line 11), i. e., whether to stop to iterate over an item due to its classification difficulty or to continue. Once the (i, f) pairs to query next have determined, HSR proceeds to obtain the vote and iterates as per the SR algorithm (line 14).

We now present a series of experiments that, besides assessing the validity of HSR, help us understand its robustness as parameters of the problem and of the algorithm change. Besides comparing it with crowd-only classification, we also compare with ML-only classification. For this baseline comparison we leverage a Naive Bayes classifier, where machines are considered to be independent and where their votes are weighted by the estimated classifiers accuracy $A_c = \{a_c\}$ (by means of tests T). The ensemble is therefore defined as follows:

$$Class(l \in \{0, 1\}) \propto \prod_{c \in C} a_c^{\delta(l,l_c)} \cdot (1 - a_c)^{1-\delta(l,l_c)} \tag{4.6}$$

where $\delta(l, l_c) = \begin{cases} 1 & l = l_c \\ 0 & l \neq l_c \end{cases}$ is the Kronecker delta function[2], and l_c is a label from a classifier c.

4.5 Analysis and Experiments

4.5.1 Metrics and Baseline experiment

The approach to analysis is based on both simulations and crowdsourcing experiments. The simulations are interesting because they allow us to understand the behavior of the approach under very different conditions, and assess the impact of the variation of each parameter, be it a parameter that describes the nature of the problem (such as the filters' power and difficulty) or a parameter of the algorithm, such as the classification threshold $\overline{P_{out}}$.

The crowdsourcing experiments allow us to get actual values of parameters for real scenarios and assess validity of results in that context - besides understanding a set of nuances important in the setup of crowdsourcing tasks. We also leverage the data to build classifiers using commonly available techniques and use their accuracies to get a feel for the kind of accuracy we can achieve for the problem at hand.

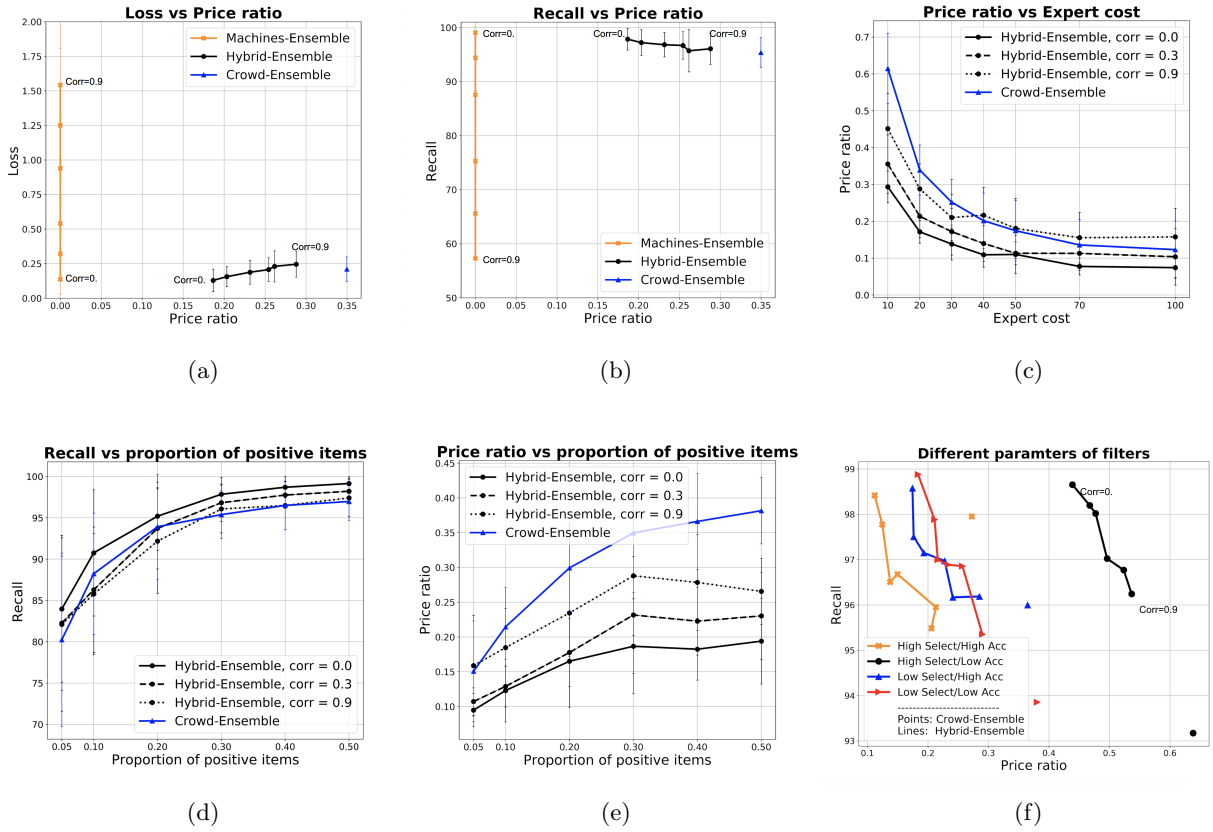
Metrics. In the experiments there are three metrics we want to assess:

[2] https://en.wikipedia.org/wiki/Kronecker_delta

1. The *loss*, computed as defined earlier, that quantifies our error weighted by how "severe" false exclusions are considered (we can similarly use F_β for this, though we choose loss in line with previous literature and also as it is more intuitive).

2. The *recall*, which is as usual defined as *true inclusions / (true inclusions + false exclusions)*

3. The relative cost of crowdsourcing, defined as the *price ratio* of the cost of crowd classification versus expert classification.



(a) (b) (c)

(d) (e) (f)

Figure 4.1: Analysis of the hybrid algorithm as key parameters vary. Settings and description are provided in the text. Crowd-Ensemble refers to the SR algorithm, while Hybrid-ensemble is HSR.

The rationale for using price ratio as a metric when comparing algorithms is as follows: we consider that the price of crowdsourcing is not just represented by the total number CV of crowd votes asked: In fact, for each false inclusion FI (for each item the crowd fails to screen out) we need to incur in the expert screening cost, as we are leaving the item on

the expert's desk. If we fail to account for this cost we overestimate the performance of the algorithm (incidentally, this is true for SR as well, although this is not discussed in the original paper and price ratio is not used there as a metric). We therefore compute the crowd cost as $CC = CV + FI * ec$[3] and compare it with what the total expert cost $EC = |I| \cdot ec$ (recall that I denotes the set of items to be classified) would be if experts did all the classification. The price ratio is then CC/EC. Notice that considering price ratio and loss also incorporates precision: lack of precision (false inclusions) means higher loss and higher classification cost (higher price ratio). For this reason we do not plot precision in the following.

To describe the results, we first describe the baseline parameter settings for the experiments, and then show how results changes as we vary each parameter.

Baseline experiment. The baseline experiment settings consists of a problem where we screen items via 4 filters, and where 30% of the items survive the screening - if we classify them correctly. We simulate 10 ML classifiers with accuracy randomly selected from a 0.5-0.95 range and on which the algorithm assumes no prior knowledge. We screen them with T=50 tests, work the math with the Beta distribution as described earlier, and keep the classifiers with 0.95 probability of having an accuracy greater than 0.5. The classification threshold $\overline{P_{out}}$ is set at 0.99, the loss ratio k in the loss function is set to 10, and the expert classification cost ec is set to 20 times the crowd label cost. This latter value is estimated from [Mortensen et al., 2016] for the case of SLRs. Notice that expert screening cost is *per item*, while the unit label cost is *per label* on an (item, filter) pair. We stress again that users only have to set parameters corresponding to their requirements in terms of loss function, and the algorithm works out the rest, including the expected price ratios and recalls corresponding to given thresholds $\overline{P_{out}}$.

In Figure 4.1a we compare the results of applying machine only (ensembled with Naive Bayes), crowd (with SR), and HSR to simulations of classifications for 1000 items. The charts plot the mean loss by price ratio per item, obtained by averaging the results of 50 iterations. Vertical bars denote standard deviation. The different dots of the machines and hybrid curves denote different values for *error* correlation among machines, simulated at 0, 0.2, 0.3, 0.5, 0.7, 0.9. As we can see (Figure 4.1a), for a similar loss level, hybrid algorithms significantly outperform the crowd in terms of both price ratio and loss. Not surprisingly, price ratio and loss worsen as error correlation increases, as ML classifiers tend to agree when they make mistakes. However, even at high correlations we maintain performances that are superior to those of crowd-only. As we do not consider crowd correlation, for the

[3]There are cases where the "expert cost" cannot be easily computed as the cost of expert labor: for example, a quasi-exact medical screening test may present practical challenges and even risks. In these cases, either domain experts can map these considerations into an "expert cost" metric, or the analysis needs to focus on precision, not price ratio.

crowd we just have a point in the chart. ML-only classification is competitive only with independent and accurate classifiers, and their performance quickly deteriorates even at small level of error correlation, as we do not assume we have strong classifiers. In the case of SLRs, with expert accuracy in the 95-98% range [Krivosheev et al., 2018b; Mortensen et al., 2016] and assuming that the items that survive the screening are in the 10-30% range, the loss lies within 0.04 and 0.18 (with the settings of the experiment being closer to the 0.18 mark, slightly better than the result of HSR but a much higher cost).

Similarly, Figure 4.1b shows the recall vs the price ratio. Recall is independent of the loss parameter k and gives us a direct indication of our capacity to identify the items that pass the filters for a given monetary investment. Notice that we disregard here the cost of obtaining the base classifiers, which, if they are built from scratch for this specific SLR, needs to be factored in when estimating price and assessing the best strategy. On the other hand, if built for the specific problem they are also likely to have higher accuracy. We do not discuss this further as again building the base classifier is orthogonal to our goal here. ndependent classifiers, (which we do not consider realistic), and deteriorates even at low correlations. We also observe that recall is high "by design" in that we fix the threshold $\overline{P_{out}}$ for considering an item as screened.

4.5.2 Effect of changes to the parameters

We now explore how results changes as we change either parameters that describe the nature of the problem (e. g., filter power, crowd accuracy, or loss and expert cost ratios) and the behavior of the algorithm (e. g., decisions on how many test data to obtain, or thresholds to set). Here we focus on a few interesting variations, and refer the reader to our GitHub repository[4] for additional details, data and source code. Notice that HSR is *adaptive*, that is, it estimates the parameters of the problem and changes its behavior accordingly. For this reason, we expect the recall to remain high, while price ratio may change.

Fig 4.1c plots how the price ratio improves with the expert cost *ec*. Recall (not plotted here) is essentially constant. For example, for a 0.5 correlation, it stays in the .96-.97 range for all *ec*. The lines flatten for high *ec* and asymptotically reach the minimal price ratio, which is the proportion of false inclusions.

Figures 4.1d and 4.1e show instead how recall and price ratio change with the proportion of items that should pass the screening (positive items). Specifically, in the chart we show the behavior when the power of one of the four filters grows. The figures show that the performances worsen in high power situations. This is because high powers mean a prior $P(i \in OUT)$ that is very close to the $\overline{P_{out}}$ threshold, and as such it is more sensitive to

[4]https://github.com/Evgeneus/crowd-machine-collaboration-for-item-screening

errors from the crowd in the first few votes. This is not a problem of HSR per se but is inherited from SR. To correct it, it is sufficient to "tone down" the prior when the power estimate is too high. Indeed, we observed experimentally that this can be achieved by structurally underestimating power by approximately 20% of its value.

Finally, Figure 4.1f shows how recall varies in the presence of one criteria that heavily differs from the other in terms of difficulty (accuracy) and power (selectivity of filters). As expected, recall remains constant because the algorithm adapts to the characteristics of the problem. Price ratio worsens as we go from an easy and powerful criteria to the most difficult case of low power, low accuracy filter where we have many incorrect votes, and even correct ones do not help us much because the screening power of this filter is low, which means we anyways need to query the other filters. HSR results are robust to variations in the number of tests and in the confidence thresholds for keeping a ML classifier (not shown). Assuming a set of classifiers in the 0.3-0.7 range instead of 0.5-0.95 cuts approximately in half the gain with respect to SR.

4.5.3 Experimental data and Experiment design

We experimented both by using existing datasets and by running crowdsourcing experiments.

We first took an existing SLR dataset by Wallace and colleagues [Mortensen et al., 2016]. This includes over 20000 crowd votes over 4000 papers, with four filters. We also run a set of experiments on Amazon Mechanical Turk and Figure Eight, with different designs, but specifically borrowing the design from Chapter 3 (Figure 3.3) for a fair comparison, and eventually built our own platform on top of Amazon Mechanical Turk for increased flexibility and for being able to plug in the adaptive algorithms into the task assignment logic. Specifically, the basic design included classifications of 100 papers for an SLR in the social informatics field, with three filters of different difficulty (one of them very difficult, with a worker accuracy barely above 0.5). The filters were: "does the paper describe a study on 65+ older adults", "does the paper describe a study that uses technology", and "does the paper describe an *intervention* type study" (the difficult one). Workers with historical overall Amazon Mechanical Turk accuracy over 70% were invited to participate. A worker would see a description of one criterion, along with a positive and a negative example. They would then proceed to labeling papers based on that criterion. The choice of a per-filter approach (asking for labels on many items for one filter) as opposed to per-item, is because it takes time to properly explain and understand a criterion (for example, explaining what an intervention is may be tricky and requires the worker to go through positive and negative examples carefully). We screen workers with two test questions and consider the labels from the remaining workers. After initial experiments

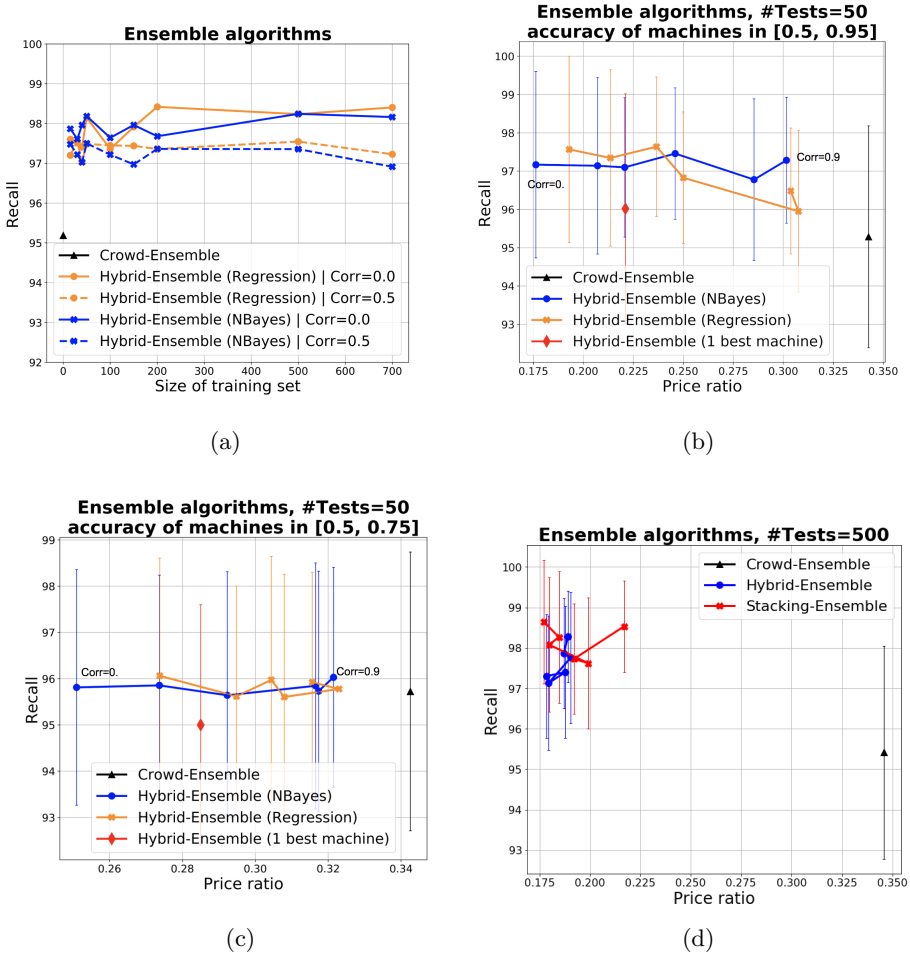


Figure 4.2: Experimental results for basic and stacking models. (a): recall for SR (black), basic HSR (blue) and HSR where classifiers are combined with regression (yellow) as the size of the training set used for the regression model varies; (b) and (c): recall vs price ratio for different distribution of classifiers accuracies; (d) results based on parameters obtained from Amazon Mechanical Turk experiments. The dots in figs b, c, d present the correlation level of machine classifiers. Corr is [0, 0.2, 0.3, 0.5, 0.7, 0.9] from left to right.

to estimate the time taken by workers, we tuned parameters to arrive at a pay rate of 10USD/hour. We obtained 10 labels for each item and filter for a total of 3000 crowd votes from 147 workers.

We then built classifiers for each filter using a variety of techniques (from KNN to random forest, variations of naive Bayes and logistic regression and others, with different kinds of document representations) and different sizes of training data to get realistic information on classifier accuracies and correlations. We obtained classifier accuracies in

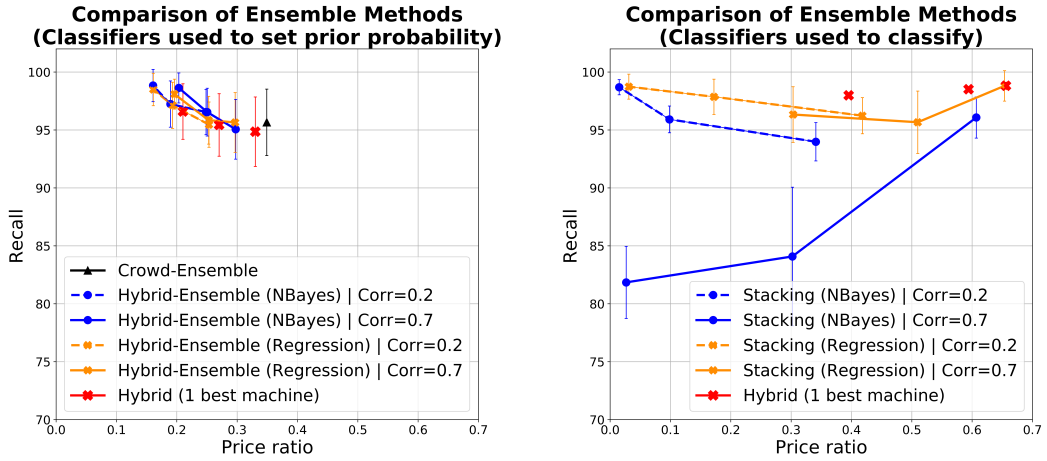
the 0.5-0.8 range, correlations in the 0.2-0.9 range, and crowd accuracy in the 0.55-0.8 range. In this case, classifiers were obtained by training on the same SLR data using 20% of the data, where the training set is obtained via crowdsourcing, so the savings for the experiment refers to the remaining portion of the items to be screened.

We then use these parameters obtained from experiments to fuel various experiments and repetition. We show the representative result in Figure 4.2d where the blue line represent HSR. We ran experiments with different combinations of base classifiers, having different correlations among them as usual represented by dots in the line. Notice that the standard deviation is very high in proportion to the scale of the chart, though we can see the impact of the hybrid approach. Higher correlations (moving from top to bottom) lead to slightly lower recall. Incidentally, we also report that, consistently with expectations, errors made by HSR and by us (the experts, in this case) were about the same: half the time we disagreed with the crowd, either the crowd was right or the decision was questionable.

4.5.4 Stacking classifiers

In the experiments above we naively included ML classifiers regardless of their error correlation. However, if two or more classifiers predict the same class (and especially if they make the same errors) then it does not help to pool their predictions - worse than that, when they make mistakes they make so by consensus thereby increasing our confidence in the incorrect prediction. There are essentially two approaches to deal with this: the first is to filter out highly correlated classifiers. If we denote with c^j_{err} the event corresponding to classifier c^j making an error, then we look to exclude classifiers where $P(c^j_{err}|c^k_{err})$ is high, and particularly above 0.5, otherwise we reinforce the error of c^k. This can be done again in the test phase similarly to what we do for estimating accuracy, for example again by assigning a Beta prior to this conditional probability. However, since errors might be rare, estimating error correlation requires more data points. If the initial (expert-provided) gold dataset is small, then we can proceed with HSR to get data points, initially with one or few ML classifiers, and then add more as we are confident of their low correlation.

However, we can also *stack* the ML classifiers by learning a model, and let this model filter or downweight correlated classifiers. Figure 4.2(a-c) show the performance of a logistic regression model built over the base ML classifiers and leveraged in HSR, compared with base HSR as from the previous section (blue) and with crowd-only ensemble (SR). The plot assumes 5 ML classifiers with accuracies as before, and a varying size of training dataset. In theory, the regression naturally copes with correlation and in the presence of a sufficiently large training set, it identifies how to effectively combine classifiers. As above, the training set can be progressively obtained via SR, and we can use base classifiers to



(a) (b)

Figure 4.3: Comparison of NB and regression ensembles when used as prior (left) and to classify items directly (right). The dots in charts represent different accuracy range of machine classifiers ([0.65, 0.75], [0.75, 0.85], [0.85, 0.95]), increasing from right to left. Ensembles have been trained over a dataset of 500 items.

identify a dataset that has higher possibility to be balanced, to get a better training set.

The charts show that in practice stacking with regression offers no improvement with respect to the naive aggregation of classifiers, and improvements are also limited with respect to taking the single best classifier in the set as opposed to an ensemble (but notice that SR takes us already to 0.95 recall, so improving from that is challenging).

We found this result surprising, although consistent with the findings briefly mentioned by [Džeroski and Ženko, 2004], where however the problem is not discussed besides the mentioning of this observation. To get clarity on this matter we investigated it further. Specifically, besides experimenting with classifiers of varying accuracies and correlation, we engaged to understand if the lack of effect is due to the specific way we use the output of classifiers - that is, to set prior probabilities (which in turn determine how to query the crowd for that item based on the HSR algorithm) as opposed to directly take a classification decision.

The results are shown in Figure 4.3. Figures a and b respectively show the performance of the ensembles when used as prior (Figure 4.3a) and when used directly to take final classification decisions, without resorting to the crowd (Figure 4.3b). The different dots denote accuracy of the base classifiers, with recall and price ratio improving with accuracy as expected (with one interesting exception discussed later). The two figures are shown in the same scale to facilitate comparison. We adopt the price ratio metric also for Figure

4.3b since classification errors mean that we incorrectly leave items for experts to classify, and this incurs in an unnecessary cost with respect to classifiers with perfect precision. Figure 4.3b shows that when we use the ensemble to make predictions, in low correlation conditions both Naive Bayes (NB) and regression ensembles are superior to the single best classifier especially in conditions of higher accuracy of the base classifiers (the difference in recall is small and within a standard deviation, while the benefit in terms of price ratio is high). With high correlation, regression ensemble and best classifiers offer similar performances, while NB drops significantly in recall, our main target metric. This is not surprising since in conditions of high correlation, ensembles with NB essentially reinforce the errors. Indeed, as accuracy of base classifiers increases so does our confidence in their vote (and in the ensemble). Specifically, often this confidence will exceed our confidence threshold $\overline{P_{out}}$ for classifying items as out directly, which explains the drop in recall for the NB ensemble as accuracy increases. In conditions of lower accuracy we take NB ensemble's predictions more cautiously, which explains the better recall (at the expense of price ratio). The benefit of regression however only manifests themselves once we have sufficient training data - in our experiments we started to observe a difference starting from training sets of 400 items.

When we leverage ensembles as prior (Figure 4.3a) as per HSR algorithm, these differences smooth out considerably and the points in the charts become closer. In HSR we never classify based on machines only, as we ask at least one vote from the crowd, which helps avoid such drops in recall. Ensembles improve over crowd-only approaches as discussed, even with weak classifiers, but because taking classifiers' results as prior is conservative and because classification threshold are high, recall errors from classifiers have a chance to be corrected by the crowd.

4.5.5 Conclusion

The main result of this chapter is a method for multi-filter classification that combines ML and human classifiers to achieve high level of classification accuracy for unit of cost. We believe the main benefit lies in the ability of the algorithm to be robust to both the characteristics of the problem (such as filters that are hard for the crowd to classify) and to weak ML classifiers. In both cases, the algorithm works out to leverage the filter, items, and ML classifiers (over specific filters and items) and builds models of classifiers to make the most of what can be screened automatically or with the crowd, leaving the rest for other classification methods (such as expert classification) without compromising on quality. Furthermore, the experimental results also helps us understand, in case we do have to build classifiers because we cannot transfer them from similar learning problems, what is the accuracy we should look for to get a benefit from hybrid classification. While

we focused on hybrid algorithms we also uncovered improvements to multi-filter crowd classification algorithms such as the opportunity to smooth the prior in high-power filter scenarios.

Although the evaluation has focused on SLR we see the approach as applicable to many multi-filter screening problems. In addition, we see active learning for ensembles in finite pool contexts such as the one described here as an interesting research problem, which is complicated by the likely presence of highly imbalanced datasets and by needs that are often conflicting, that of leveraging the crowd efficiently to screen items out but to also get training data classified as in. Another interesting thread of analysis is to see if crowd workers can, besides labeling (item, filter) pairs, actually identify features that machines can then leverage for the problem at hand.

Finally, an interesting problem is the extent to which learning can be transfered across finite pool problems of the same kind (e. g., different SLRs), especially in terms of identifying general approaches and methodologies that can be adopted in different domains.

Chapter 5

Balancing Learning vs. Exploitation Trade-off in Active Hybrid Classification

In this chapter, we focus on *screening* problems where we look for items satisfying a conjunction $p_1 \wedge p_2 \wedge \ldots p_n$ of predicates within a *finite pool* of candidate items. Screening problems occur frequently in practice [Mortensen et al., 2016; Krivosheev et al., 2018b; Franklin et al., 2011; Lan et al., 2017]. For example, we may want to look for *kids-friendly* hotels *close to subway stations* given a pool of hotel descriptions, or for scientific papers that measure *happiness* in *older adults* given a set of candidate papers obtained via keyword search.

Arguably, no currently available search engine is able to answer such queries with generally applicable approaches (e. g. keyword search). Unless a specific screening logic based on domain-specific knowledge can be implemented, there are essentially two ways to screen items of interest: i) train and exploit machine learning (ML) algorithms or ii) resort to human classification (via experts or by crowdsourcing).

Research in crowdsourcing has made impressive progress in the last few years, and the crowd has been shown to perform well even in difficult tasks [Callaghan et al., 2018; Ranard et al., 2014]. However, crowdsourcing remains expensive, especially when aiming at high levels of accuracy, which often implies collecting more votes per item to make classification more robust to workers' errors.

ML algorithms have also been rapidly improving. However, in many cases they cannot yet deliver the required levels of precision and recall, typically due to a combination of difficulty of the problem and (lack of) availability of a sufficiently large dataset. Consider for example the problem of screening scientific literature: ML works very well for retrieving

papers that describe *randomized controlled trials* (RCT) [Wallace et al., 2017b], also thanks to the availability of very large training datasets. Instead, when looking for papers measuring happiness among older adults, ML approaches are challenged both by the lack of training data and by the intrinsic difficulty of the problem: papers may not even mention the term *happiness* but only related constructs, or again happiness may be discussed but not measured.

Hybrid classifiers (HCs) are rapidly emerging as a way for ML and crowds to join forces to improve accuracy and reduce costs beyond the traditional process of labeling data, training a model, and then using a model to classify items automatically (see e. g. [Imran et al., 2016, 2014]). The underlying idea is that ML algorithms, even when weak for the problem at hand, can still be exploited. For instance, they can be used to reduce the workload of the crowd by labeling items for which they have high confidence, or they can contribute to the classification decision (e. g., by casting a vote, just like crowd workers do) [Vaughan, 2017; Kamar et al., 2012]. For example, in medical diagnosis, crowd and machine have been shown to be able to jointly screen and select cases to be brought to the attention of medical doctors [Callaghan et al., 2018] with impressive accuracy.

In this chapter, we are specifically interested in cases where no pre-existing trained ML model is available. Under such circumstances, *active learning* (AL) techniques can be leveraged to optimize the cost/quality trade-off in the learning process. However, at some point through the AL process we can decide to stop collecting training data (in a way, stop *learning*) and use the remaining budget to *exploit* the learned model by applying hybrid classification (or simple ML classification if the model accuracy is sufficient).

As we will show later, learning and exploitation strategies are often at odds. In other words, the *active learning* policy and the *active crowd classification* policy (selecting remaining items in the pool with the goal of classifying them efficiently) may prioritize different items. In the following we refer to the balancing of active learning and active classification in finite pool contexts as *active hybrid classification* (AHC). What makes the problem interesting is that when we approach a new task, in general we do not know how well ML or crowd can perform and what is the learning curve of the given ML classifiers. The available budget and pool size (which are instead typically known) also come into play when deciding an AHC policy.

Problem Formulation. Given a crowdsourcing task, a set of ML algorithms (which we assume to be initially untrained, for a specific classification problem) with their *active learning* policy, and a crowd classification algorithm with its *active classification* policy, our goal is to identify how to balance the learning versus exploitation trade-off to improve the overall classification cost and quality.

Original Contribution. In this chapter, we contribute a formulation for the general

problem of active hybrid classification. We propose a generic approach for AHC (independent of the specific ML or HC algorithm adopted) and show that striking the right balance between learning and exploitation can have a significant impact on classification cost and quality. We then propose and discuss several budget allocation heuristics that can be adopted to efficiently balance learning and exploitation. Such heuristics are assessed via a set of experiments on datasets with diverse characteristics to establish their validity (or lack thereof) in different contexts, and ultimately derive some rules of thumb of wide applicability.

5.1 Background

5.1.1 Active Learning

Active Learning (AL) studies how to select items to improve the learning rate of ML classifiers with respect to random selection [Aggarwal et al., 2014]. One of the most popular active learning strategies, dating back to the 1990s, is *uncertainty sampling* [Lewis and Gale, 1994], where manual labeling is sought first on items for which the classifier is less certain. [Mozafari et al., 2014] proposes to also consider items that have the biggest impact on the learned model if the classifier is found to be wrong in the prediction. Interestingly, this paper also discusses hybrid classification and the issue of when to stop collecting votes to train the classifier. However, it deals neither with finite pool problems where there is a trade-off between learning and exploitation, nor with screening problems.

Abundant research has since proposed a number of different approaches and optimizations, and recent studies have shown that active learning can be successfully exploited in deep networks as well [Shen et al., 2017]. Although the problem we deal with here has aspects of active learning - since we also identify how to prioritize items to label, our goal is to balance learning vs exploitation, and the item prioritization is then made by the ML or HC algorithms based on their own logic. This being said, we do show in the experiments and analysis section how the choice of specific AL policies affect the performance of hybrid classifiers.

5.1.2 Learning to Learn

The problem of balancing learning and exploitation in HC contexts can be cast as a *multi-armed bandit* (MAB) problem (see [Lattimore and Szepesvar, 2019] for a wonderful introduction to MAB). [Baram et al., 2004] study how, given an ensemble of AL algorithms, we can choose the items on which to obtain gold labels for training, reassessing the decision after a round of N items. The problem is mapped to an *adversarial*, contextual MAB

problem where the items to be selected are the *arms* of the bandit (that is, the alternative choices at our disposal) and the different AL algorithms are the *experts*, each of them indicating a different set of items (for example, uncertainty sampling vs random sampling). They empirically show that by extending *Exp4* [Auer et al., 2002] - a popular algorithm for adversarial MAB - with a method for estimating the AL reward they can identify the best AL algorithm in the ensemble for different datasets - and even outperform the single best algorithm. [Hsu and Lin, 2015] extend this result following a similar approach but leveraging an improvement of Exp4, called *Exp4.P*, which achieves regret bounds with high probability [Beygelzimer et al., 2011]. They also adopt a reward function to measure the performance of the trained classifiers based on importance-weighted accuracy [Ganti and Gray, 2012], and show that results often outperform previous approaches.

Our problem is similar in that we have to choose between alternatives in an explore vs exploit fashion typical of MAB problems, and as in the research above we cannot assume i.i.d. rewards for arms. However, we observe that we do have expectations on the specific shape of ML and CC curves which neither give i.i.d nor adversarial rewards in the strict sense, but tend to follow learning/exploitation curves, although noisy ones and with local minima. We can therefore exploit this knowledge. Furthermore, we cannot adopt the notion of rewards as suggested in the literature above, both because we do not deal only with active learning arms and, even for the active learning side, the contribution of the additional training is not in terms of increased accuracy but rather on how this increased accuracy impact the crowd classification cost/quality trade-off for the items left to classify in our finite pool. We will get back to this point when discussing approaches to tackle the screening problem.

5.2 Method and Problem Formulation

Figure 5.1 shows the high level architecture of the proposed Active Hybrid Classifier (AHC). AHCs processes items in iterations consisting of three phases (denoted by dashed rectangles in the Figure): i) selecting a batch of items and predicates to submit to the crowd for voting, ii) obtaining the votes (performing the actual crowdsourcing task), and iii) processing the votes by first re-training ML classifiers with the new information and by then leveraging the ML contribution to guide the crowd classifier. Batches are processed until either all items have been classified or the budget has been exhausted - at which point items for which no decision could be taken are left for expert or ML-only classification.

We already discussed why we need AL to efficiently train the ML classifier. Interestingly, we face the same item (and predicate) selection problem in crowdsourced classification as well. We refer to the problem of item (and predicate) prioritization for crowd classification

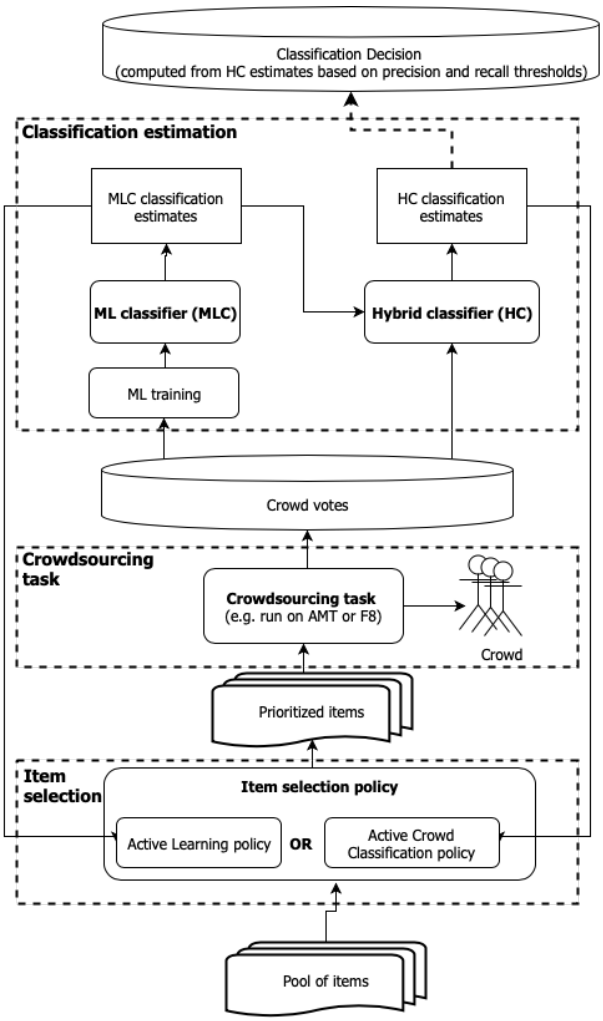


Figure 5.1: Architecture of the Active Hybrid Classifier.

as *active crowd classification*, for symmetry with active learning. To see this, consider again the hotel search example: humans can easily screen hotels based on location by glancing over a map, thereby filtering out the majority of items and leaving a much smaller number of hotels on which to assess the more complex *kids-friendly* predicate. Therefore, we can choose to first poll the crowd only on location. Then, for those hotels that pass the location screening, we can obtain votes for kids-friendliness. This approach can be adopted in any screening problem, and is followed in query optimization as well [Hellerstein and Stonebraker, 1993].

The challenge in this scenario is that *active learning and crowd active classification are often at odds*: they can select different items and predicates for crowd voting. Indeed, the active classification approach based on predicate selectivity mentioned above is particularly "bad" for training ML algorithms: not only it does not follow active learning practices such

as uncertainty sampling, but, by focusing on eliminating items efficiently, it creates an extremely unbalanced dataset (among other possible sampling biases). This means that we may be facing a *learning* vs *exploitation* trade-off, i. e., whether we invest budget to train ML efficiently (selecting items via *active learning*), or to exploit the trained algorithms and classify items in our pool (selecting items via *active crowd classification* algorithms and using HC to classify). Intuitively, if the pool is small and the problem is hard for ML, it may be convenient to lean towards exploitation. If the problem is easy for ML and the pool is large, focusing more on learning may be the best strategy.

Problem Formulation. The problem can be formulated follows. *Given a tuple* (I, PR, A, B, Q) *consisting of:*

- a finite pool of items I to be screened based on a conjunction of predicates $PR = (p_1 \wedge p_2 \cdots \wedge p_n)$,
- a set of algorithms $A = (A_{ml}, A_{al}, A_{hc}, A_{ac})$ which includes a ML classifier A_{ml}, an active learning algorithm A_{al}, a hybrid classification algorithm A_{hc}, and an *active crowd classification* algorithm A_{ac}.
- a quality/cost trade-off goal Q, and a budget B

we aim to identify a decision strategy that defines how to balance *learning* – i. e., selecting items with the goal of improving A_{ml} efficiently, thanks to active learning A_{al}, versus *exploitation* – selecting items via A_{ac} with the goal of reaching a classification decision efficiently, also thanks to the contribution of the (partially trained) classifier A_{ml}.

We observe that rather than having one ML algorithm for the overall screening problem, it is often convenient to have a separate classifier A^p_{ml} for each predicate p. There are several reasons for this: besides being a requirement of some HC algorithms, having separate classifiers facilitates their reuse [Cherman et al., 2016]. For example, in literature reviews it is not uncommon to revisit definition of predicates (e. g., we realize that "older adults" is ambiguous and we want to be more specific, such as "adults past their retirement age in their country of residence") while keeping the others unchanged, or to apply the same predicate to a different review (many reviews may need to focus on older adults).

Furthermore, crowdsourcing tasks focused on a single predicate may offer better results when the predicates are complex to understand and training is required. We therefore tackle the general multi-predicate case and use A_{ml} to denote a *set* of ml algorithms $\{A^p_{ml}\}$ in the following.

The goal Q typically involves tackling a cost-accuracy trade-off, setting a threshold on recall, precision, or F_β measures (where β skews the F-measure to favor precision or recall), or some other cost function [Mortensen et al., 2016]. The precise form in which it is expressed is not particularly relevant to our discussion, and in the following we simply take F_β and budget spent as measures.

AHC Approach and Algorithm The idea behind the approach is to proceed in iterations, where at each run we go through the phases described in Figure 5.1, having selected active learning or active crowd classification at phase 1. Borrowing notation from research in multi-armed bandits, we use $t = 1, 2 \dots n$ to refer to iterations and $K = (k_l, k_e)$ to refer to the possible actions (arms) that can be undertaken at each iteration, which are limited to *learning* (denoted with k_l, meaning that we select items via A_{al}) and *exploitation* (denoted with k_e, meaning that we select items via A_{ac}). At each iteration t we observe a context $c_t = (\Psi_t^{ml}, \Psi_t^{hc}, CI_t, UI_t, b_t)$, where:

- $\Psi_t^{ml} = \{\psi_{t,ml}^{i,p}\}$ are the predictions, expressed as probability of item i satisfies predicate p, as estimated by A_{ml}^p trained over the labels L_t
- $\Psi_t^{hc} = \{\psi_{t,hc}^{i,p}\}$ are the predictions, expressed as probability of item i satisfies predicate p, as estimated by $A_{hc}^{i,p}$ that takes as input labels L_t and Ψ_t^{ml}
- CI_t is the set of items for which A_{hc} has reached a decision so far (depending on classification thresholds)
- UI_t is the set of items on which A_{hc} is still undecided. It is always true that $UI_t \bigcup CI_t = I$
- b_t is the budget used so far, out of a total budget B

Notice that classification decisions are *only* made as a result of applying A_{hc}. This does not mean that asking for crowd votes is always required: when the budget is exhausted, for instance, A_{hc} can decide to leave items unclassified or to trust the ML prediction. The choice here depends on the specific HC algorithm used, and AHC is agnostic to that. Given this algorithm, we now discuss policies to tackle the learning vs exploitation trade-off.

5.3 Approach and Heuristics

The problem of selecting which algorithm (A_{al} or A_{ac}) to adopt at each iteration for prioritizing items can be cast as a multi-armed bandit problem, where we balance exploration of the effectiveness of each arm and then exploit the findings to classify items efficiently. This is indeed the approach followed in active learning when choosing among different ML classifiers and active learning algorithms to train them. The problem is cast as an adversarial bandit with expert advice [Baram et al., 2004; Auer et al., 2002; Beygelzimer et al., 2011], where the policy defines the probability distribution over arms $[K]$ based on weights associated to such arms. As an example, adopting the popular Exp3 algorithm [Auer et al., 2002] and adapting it to our context would entail:

1. Defining and estimating the value (*reward*) $\hat{x}_k(t)$) obtained as a result of havign chosen arm k at iteration t. In our context a meaningful reward can be the reduction in the expected cost for classifying all items in the pool.
2. Assigning weights w_{k_l} and w_{k_e} to the arms as follows:

$w_k(t) = w_k(t-1)e^{0.5*\gamma\dot{\hat{x}}_k(t)}$, where γ is the probability we should assign to random (uniform) selection among arms, regardless of what the weights say. The weight for the arm that has not been chosen remains unchanged.
3. Determining the probability of choosing arm k (e. g., the probability of choosing A_{ac}) based on Equation 5.1.

$$P(k_l) = (1-\gamma)\frac{w_{k_l}(t)}{w_{k_l}(t) + w_{k_e}(t)} + 0.5 * \gamma \tag{5.1}$$

This approach also has specific upper bounds in terms of expected *weak regret* (that measures how much worse is our reward compared with the best among the two "static" policies that always choose the same arm) and the literature mentioned above shows extensions to achieve a specified regret with high probability.

While we could follow the same approach, there are three important differences between the problem discussed here and the "learning by learning" method discussed above.

First, computing the reward here is more challenging. We retain all the mentioned difficulties of estimating the accuracy of A_{ml} in active learning contexts (the accuracy essentially is the reward in active learning) [Nguyen et al., 2015a; Beygelzimer et al., 2009]. In addition, the effect of the incremental learning performance has to be estimated in the context of its contribution to A_{hc}, since A_{ml} is not used to classify but to assist in crowd classification. This brings additional noise to the estimation. Furthermore, such contribution changes over time, because i) as A_{hc} progresses we are left with more difficult items to screen, which means that the population of items changes, and ii) arms are not independent: pulling the *learn* arm affects the reward we get in the future by pulling the *exploitation* arm. Finally, classes in nearly every screening problem are highly unbalanced in favor of exclusions. This makes not only harder to train algorithms, but - most importantly for our estimation problem - makes it hard to assess accuracy in an active learning context unless a large pool of labeled items is available.

Second, the incremental value of adding budget is likely to decrease as the iterations proceed. In terms of A_{ml}, learning curves generally tend to level off. At the beginning, every new example provides new information; as we progresses, it is likely that some of the information brought by the new item is already incorporated in the algorithm. This however is not a guarantee, and in fact the learning behavior can be erratic. Interestingly, the same is often true for active classification: algorithms try to greedily screen items out efficiently, focusing first on doing cheaply what it can do cheaply, and leaving items harder to classify for later iterations or even give up on them and leave them to experts.

Third, weak regret over the best arm would be a conservative measure of effectiveness: choosing always the same arm (that is, relying only on A_{ml} or only on A_{hc}) has been shown to perform *worse* in many cases than hybrid strategies [Krivosheev et al., 2018a].

For these reasons, while we see research on optimization and theoretical bounds as beneficial in the long run, in this chapter we focus on two key questions: i) whether studying the *learning* vs *exploitation* trade-off in AHC is even a problem worth studying and under which conditions benefits can be observed, and ii) whether it is possible to identify a set of "rule of thumbs" and simple heuristics that work well for a variety of problems with different characteristics. Specifically, out of the space of all possible policies Π, we look here into deterministic policies $\pi \in \Pi$ that begin with the *learning* arm, to then switch at a specific point to the *exploitation* arm. We identify the following policies π:

Baseline: The simplest approach is simply to stick to the same arm: $\pi(c_t) = [1, 0]$ or $\pi(c_t) = [0, 1]$, for any context, where the array returned by the policy denotes the probability of choosing A_{ml} or A_{hc}, respectively.

Fixed point switch. The second option identifies a priori a percentage of the items (or of the available budget) to devote to ML training (that is, selecting the ML policy) and then switching to the crowd policy. This may seem arbitrary, but experiences with datasets revealed that there are heuristics that work well in a variety of cases, such as the 70/30 or 80/20 split between training and test data for classical ML algorithms. For example, if we base the decision on setting a percentage γ of budget B for learning, then the policy becomes $\pi(c_t) = [1, 0]$ if $B_t \leq \gamma B$, and $\pi(c_t) = [0, 1]$ otherwise. Notice that while with this heuristic we could cast the problem to the well-known stopping or *secretary problem* [Bruss, 2000], in our case we do not have independent and randomly shuffled "secretaries" (rewards follow a trend) and in general all the reward estimation challenges discussed above apply here as well.

Stochastic policies (with fixed probability). In this class of policies we identify a priori a static distribution over arms $[K]$ (that is, defining the probability P_l of choosing to learn vs exploit), and at each time we select a policy based on that probability. This results in policy $\pi(c_t) = [P_l, 1 - P_l]$.

Adaptive Contextual Policies. Adaptive policies continuously estimate the characteristics of the problem at hand, and specifically the ability for the ML classifier to rapidly improve to provide useful contributions, as well as the effectiveness of crowd classification. Intuitively, we begin with the learning arm and estimate the accuracy (F_β) of A_{ml} as a result of selecting the learning arm as per the literature [Nguyen et al., 2015a; Beygelzimer et al., 2009]. We can then compute the trend in accuracy (percentage difference $\delta(t)$ in accuracy between iteration, possible smoothed via a moving average) and map values of $\delta(t)$ to a probability value k_{al} of choosing the learning arm via a mapping function m (similarly to what logistic regression does with the logistic curve). This results in policy $\pi(c_t) = [m(\delta(t)), 1 - m(\delta(t))]$.

5.4 Analysis and Experiments

5.4.1 Experiment Objectives and Design

We experiment the approach and heuristics described here with the goal of: i) *understanding whether there even is a decision problem to solve*, that is, whether selecting specific policies *can* lead to cost and quality trade-offs that are different from that of the obvious baselines (fixed policy), ii) evaluating our heuristics and identify if we can derive *rule of thumbs applicable in different contexts*, and iii) understanding which aspects of the crowdsourcing problem may impact the effectiveness of given heuristics

We identify three datasets with different characteristics in terms of accuracy that ML or crowd achieve, and in terms of predicate selectivity:

Amazon Reviews. The dataset contains reviews on Amazon products, with information about the *sentiment* expressed in the reviews and the product category [1]. We created a 5000-item dataset with labels for two predicates: 1) whether the review is on a book (`Books`) 2) and whether it is a `Negative review`. The `Books` predicate has selectivity $\theta = 0.61$, while `Negative review` has $\theta = 0.10$. If we look for negative reviews on books, therefore, only about 5% of reviews pass the screening. With this dataset, the task is relatively easy for the crowd. ML performs well on the Books predicate and less well (with the SVM algorithm we adopted) on sentiment analysis.

Medical Abstracts. This two-predicate screening dataset is taken from the OHSUMED test collection [Joachims, 1998], where p_1 is *"Does the paper study Cardiovascular Diseases?"* and p_2 is *"Does the paper describe Pathological Conditions, Signs, and Symptoms?"*. Selectivity values are $\theta_1 = 0.18$ and $\theta_1 = 0.28$ for p_1 and p_2 respectively. The dataset is highly unbalanced, with 5% of items out of 34387 abstracts passing the screening, and an harder task for ML.

Systematic Literature Review (SLR). This dataset mimics the screening phase of the SLR process, where the abstracts returned from a keywords-based search need to be screened by researchers based on specific predicates. p_1: *"Does the paper describe an experiment targeting older adults"?* p_2: *"Does the paper describe an intervention study?"* The selectivity is $\theta_1 = 0.58$ and $\theta_2 = 0.20$, and the resulting set of relevant items is 17% of the dataset. This task is hard for the crowd and even harder for ML classifiers, especially for predicate p_2 (accuracy at approximately 0.65).

We also experiment with different ML algorithms and different AL strategies (namely, random sampling and uncertainty sampling, which is the most common AL approach). On the crowd side we focus on the *Hybrid Shortest Run* (HSR) algorithms for active crowd classification and hybrid classification, as to the best of our knowledge they represent the

[1] https://www.cs.jhu.edu/ mdredze/datasets/sentiment/

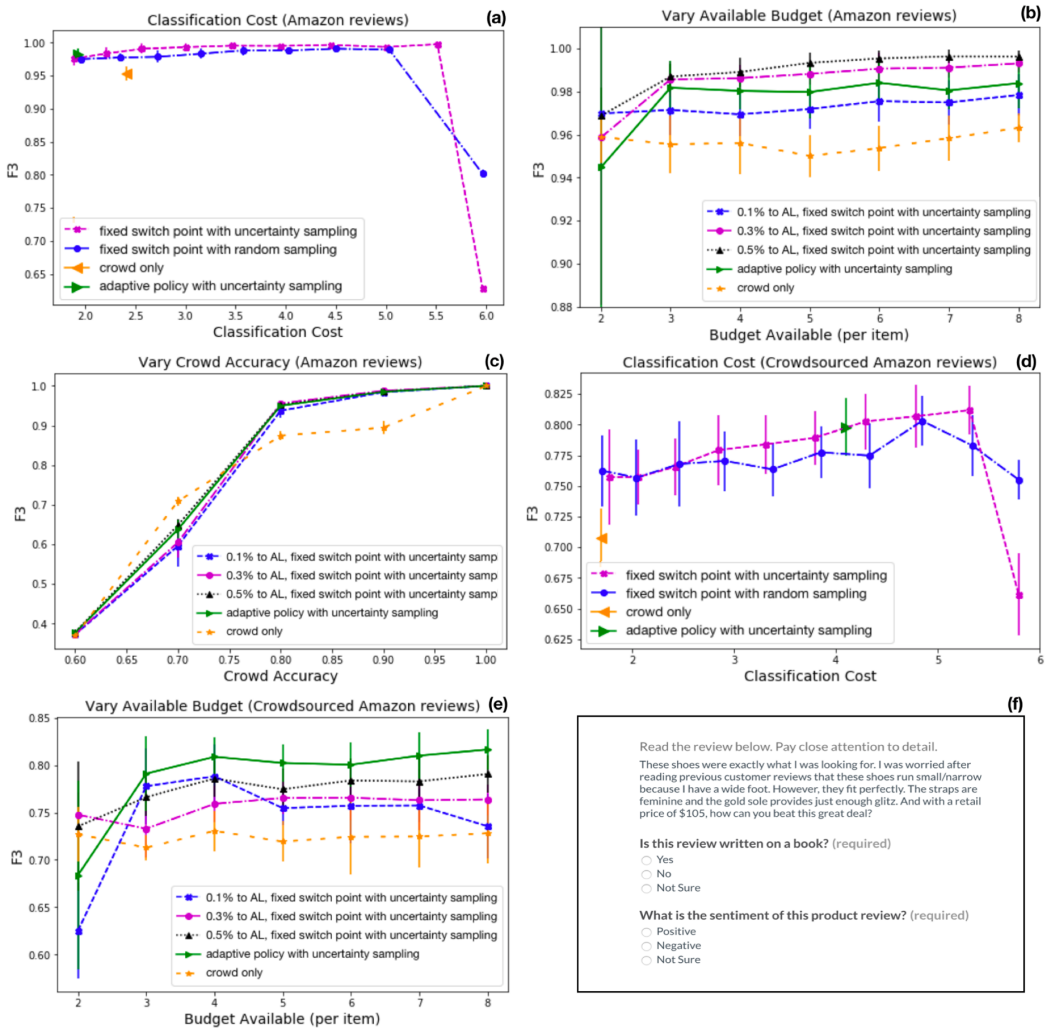


Figure 5.2: Results on the amazon reviews dataset (see text for description).

state of the art in screening problems [Krivosheev et al., 2018a]. We then test the various heuristics by training classifiers, crowdsourcing votes, and performing the AHC algorithm described earlier. In addition to the studies with crowdsourced votes, we simulate how results would change as key characteristics of the problem vary - for example, in problems with very high or low crowd accuracy - to get a better sense for the conditions under which heuristics can be effective.

5.4.2 Crowdsourcing Tasks

For Amazon and SLR datasets, we also ran tasks on the *FigureEight* (F8) crowdsourcing platform to collect crowd votes. For Amazon reviews, we collected votes for 1000 items, with 5 votes per item and predicate, obtaining a total of 10000 votes. The task is shown in

Figure 5.2(f). Workers were paid 3¢/review, with hourly pay rate >10$/hour. The initial test screening included 4 questions (100% accuracy required) and with 25% of honeypots after that. Crowd accuracy for both predicates was 0.94.

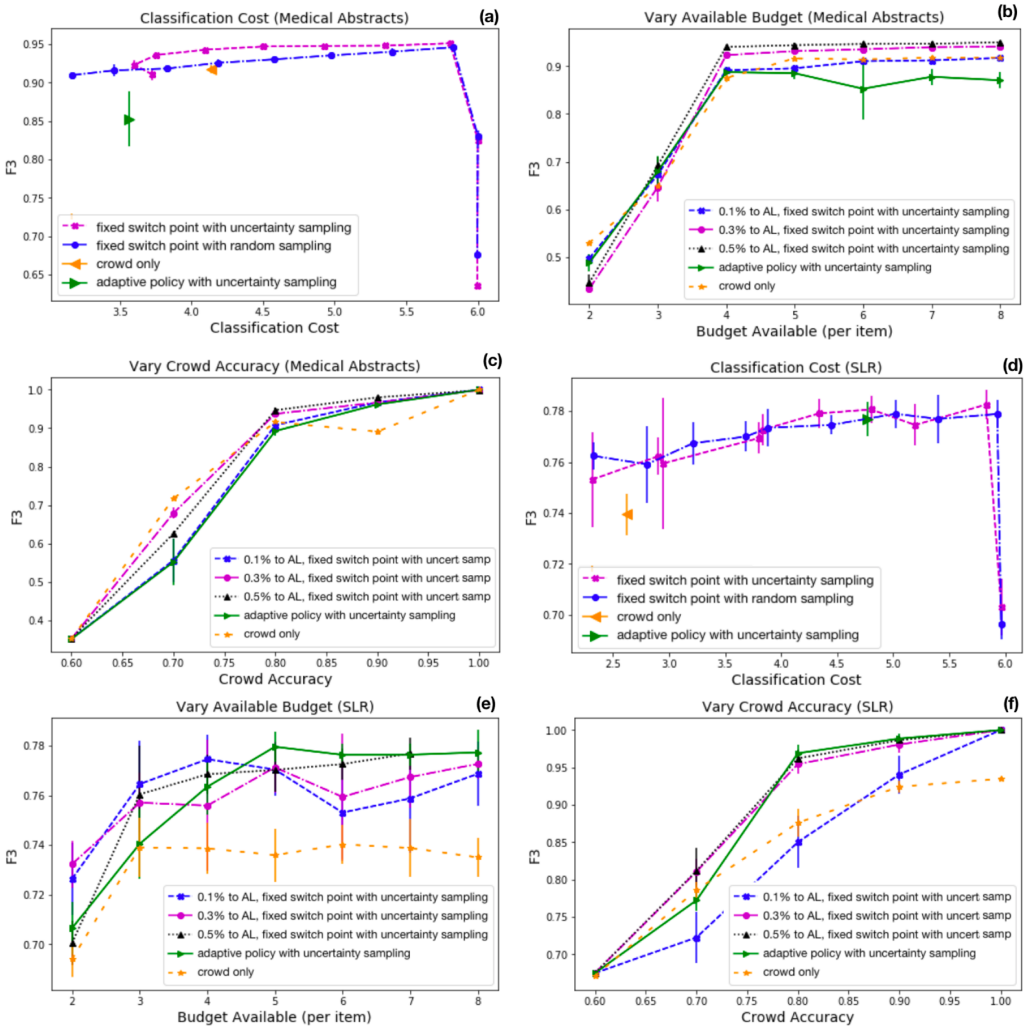


Figure 5.3: Results on the MED Abstracts and SLR datasets (see text for description).

For SLR abstracts we collected 5-7 votes per paper, limiting workers' contributions to a maximum of 18 judgments to allow for worker diversity. A total of 236 workers from 41 countries contributed with 1321 judgments. The payment was of 15 cents per vote, which again computing the reading time spent on each abstract was above 10USD/ hour. As discussed, each worker focuses and votes for one predicate only, since understanding predicates is complex and takes time. Resulting accuracies here were 0.6 (*intervention* predicate) and 0.8 (*older adults*).

5.4.3 Experiment settings, execution and results

To analyze the heuristics, we run AHC over the available datasets, training classifiers and performing hybrid classification first over simulated crowd votes and then on actual votes. All the source code, results, and datasets are available as companion material, along with charts that explore additional behaviors (here we limit to a few interesting ones).

Figure 5.2(a) shows the results of applying the budget allocation heuristics in terms of classification cost (expressed as average number of crowd votes per item) versus F_β (assuming $\beta = 3$ - but the patterns are similar for $\beta = 1$ (see the companion material)). AHC was applied to classify a pool of 5000 amazon reviews, and crowd votes were simulated but using the same accuracy values obtained from the crowdsourcing experiments (p1=p2=0.94). The orange dot corresponds to the static policy that always applies HSR's active crowd classification (no ML training and no hybrid classification - only crowd). The purple and blue lines show the baseline and fixed switch point policy for different switch points, expressed as percentage of budget devoted to the learning arm, from 0.1 to 1 (from left to right). The purple curve adopts uncertainty sampling, while the blue denotes random sampling. The green dot represents an adaptive policy that switches from learning to exploitation when the increase in accuracy falls below 2% for the first time after the first 500 items (accuracy is always erratic at first). Vertical bars denote standard deviation over 10 repeated runs of the entire AHC process, and the dots show the mean value.

The figures show that for small percentages of budget allocated to learning, AHC outperforms crowd-only classification in both quality and cost. As the learning budget increases, so do both cost and quality so that at some point the choice becomes a trade-off. However, when we approach 100% devoted to learning, the performances drop below that of crowd classification. This latter results is in line with work that separately studied performance of machine-only and crowd-only approaches [Krivosheev et al., 2018a] and confirms that even a "sprinkle" of crowd votes is crucial to classification accuracy unless machines are near-perfect. Figure 5.2(d) just below shows the same analysis but on crowdsourced data. Results are more noisy likely because the crowdsourced dataset is of only 1000 items (also, recall that in screening problems datasets are often unbalanced). With fewer items we also have worse ML accuracy, hence the lower F_β.

Figures 5.2(b) and (e) show the behavior of the same curves as the overall available budget increases, or equivalently, as the budget is fixed but the pool size decreases (budget is expressed in average number of votes available per item) for simulated and actual crowd votes respectively. Because ML accuracy here is very high, the performances remain good even with a low budget (we will see that this is not the case for other datasets). Finally, Figure 5.2(c) simulates variations in crowd accuracy. Notice that the range of values for

the F measure here is much wider. Here we notice that the benefits of hybrid classification over crowd-only begin to be noticeable when crowd accuracy goes above 0.8, as otherwise the training dataset itself becomes noisy.

Figures 5.3(a) and (d) show the same experiments run on the medical and SLR datasets, where we recall that on the medical datasets we do not have crowdsourced data (we simulated accuracy at random in the [0.6, 1] interval), while for the SLR dataset we only have simulated crowd votes and take the real crowd accuracy of 0.6 and 0.8 respectively for the two predicates. The trends are similar to the Amazon dataset, and as usual we see the larger variance in Figure (d) as for SLR we only have a small pool of items (just over 800). We still see however that for small percentages of budget invested in learning the results are pareto-optimal with respect to crowd classification. Also similar is the impact of crowd accuracy, with the benefit of AHC becoming manifest at crowd accuracies between 0.8 and 1, which are fairly common values (again notice that the difference in the F measure at 0.9 is high). For these datasets who have lower accuracy we also see that as the total available budget drops (Figures 5.3(b) and (e)) classification performances degrade significantly, as when we run out of money we use only machines to classify. We do not show here the stochastic policy because it is dominated by the fixed switch point (given that we determine a proportion of budget to learning, it is more efficient to learn first rather than later).

5.4.4 Conclusion

The results show that even a small fraction of budget invested in ML training leads to improvement in cost and accuracy with respect to the state of the art screening algorithm available. Increasing the budget allocation to learning up to 50% increases accuracy but also cost, and from that point on the improvements vary. We also observe that a difference with respect to crowd only classification become more manifest as crowd accuracy grows over 0.7-0.8 as this leads to "good" training data. In summary, adding a sprinkle of ML to crowd classification and a sprinkle of crowd to machine classification leads to very high accuracy and better results in finite pool problems, provided that the budget vs pool size ratio is such that we can collect crowd votes rather than use ML only.

While we tried to obtain diverse datasets, the applicability we claim is limited to screening problems with binary predicates, and to the HC algorithm adopted in the experiment. We also did not explore variations of adaptive policies, and we did not attempt yet to compute reward in multi-armed bandit style. All this is part of our ongoing work.

Algorithm 4: **Active Hybrid Classification Algorithm**

Input: $I, PR, B, Q, (A_{ml}, A_{al}, A_{hc}, A_{ac})$
#items, predicates, budget, quality objective, algorithms
Output:CI_t, b_t, A_{ml}

(1) $t \leftarrow 0$ *#iteration number*
(2) $UI_t \leftarrow I$ *#unclassified items*
(3) $CI_t \leftarrow \{\}$ *#classified items so far*
(4) $b_t \leftarrow 0$ *#budget spent*
(5) $L_t \leftarrow \{\}$ *#labels so far*
(6) $\Psi_t^{ml} \leftarrow \{\}$ *#machine probabilistic predictions*
(7) $\Psi_t^{hc} \leftarrow \{\}$ *#HC probabilistic predictions*
(8) $c_t \leftarrow \{\Psi_t^{ml}, \Psi_t^{hc}, CI_t, UI_t, b_t\}$ *#context at time t*
(9) $h_t \leftarrow \{c_t\}$ *#history*
(10) **while** $UI \neq \varnothing$ **and** $b_t < B$
(11) $\quad t \leftarrow t + 1$
(12) $\quad [p_l, p_e] \leftarrow \pi(h_{t-1}, Q)$
(13) $\quad k \leftarrow select\_arm(p_l, p_e)$ *#pick an arm randomly based on prob. distribution* $[p_l, p_e]$
(14) $\quad$ **if** $k = k_l$
(15) $\quad\quad I_t \leftarrow prioritise(UI_{t-1}, A_{al})$ *#prioritised items*
(16) $\quad\quad l, cost_l \leftarrow annotate(I_t, n)$ *#crowdsource top-n items*
(17) $\quad\quad L_t \leftarrow L_{t-1} \bigcup l$
(18) $\quad\quad A_{ml} \leftarrow train\_machines(A_{ml}, L_t)$
(19) $\quad\quad \Psi_t^{ml} \leftarrow predict\_prob(A_{ml}, UI_{t-1})$
(20) $\quad$ **if** $k = k_e$
(21) $\quad\quad I_t \leftarrow prioritise(UI_{t-1}, A_{ac}, \Psi_{t-1}^{ml})$ *#prioritised items*
(22) $\quad\quad l, cost_l \leftarrow annotate(I_t, n)$ *#crowdsource top-n items*
(23) $\quad\quad L_t \leftarrow L_{t-1} \bigcup l, \quad \Psi_t^{ml} \leftarrow \Psi_{t-1}^{ml}$
(24) $\quad \Psi_t^{hc} \leftarrow predict\_prob(A_{hc}, UI_{t-1}, \Psi_t^{ml})$
(25) $\quad ci \leftarrow classify(UI_{t-1}, \Psi_t^{hc}, Q)$
(26) $\quad CI_t \leftarrow CI_{t-1} \bigcup ci, \quad UI_t \leftarrow UI_{t-1} - ci$
(27) $\quad b_t \leftarrow b_{t-1} + cost_l, \quad c_t \leftarrow (\Psi_t^{ml}, \Psi_t^{hc}, CI_t, UI_t, b_t)$
(28) $\quad h_t \leftarrow h_{t-1} \bigcup \{c_t\}$
(29) **return** CI_t, b_t, A_{ml}

Chapter 6

Detecting and Preventing Confused Labels in Crowdsourced Datasets

Crowdsourcing an accurate training dataset for Machine Learning (ML) algorithms is crucial for their resulting performance. It is however challenging for many reasons, from low-quality annotators to poor task design. Abundant literature has proposed methods for addressing such challenges, from task design methodologies, to ways of improving or coping with poor task design [Chang et al., 2017; Manam and Quinn, 2018; Eickhoff, 2018; Zheng et al., 2017], worker testing strategies (to exclude low-quality annotators), and aggregation of annotations by several workers. For example, *truth finders* methods have been recently proposed [Giancola et al., 2018; Li et al., 2016] to identify the correct annotation based on dependencies between workers [Dong et al., 2009], heterogeneity of values [Li et al., 2014b], data correlation [Pochampally et al., 2014], different types of workers' accuracy and similarity [Zhao et al., 2012; Li et al., 2018; Dumitrache et al., 2018], and so on. Similarly, literature in crowdsourcing has proposed a number of algorithms to collect and aggregate crowd votes with the goal of minimizing errors in annotations, from simple majority voting to variations of Expectation Maximization [Liu and Wang, 2012; Dong et al., 2013; Whitehill et al., 2009], to algorithms that focus specifically on screening (identifying items that satisfy a conjunction of predicates) [Mortensen et al., 2016; Krivosheev et al., 2017; Lan et al., 2017].

However, a particularly subtle form of errors is due to *confusion* of observations, that is, crowd workers (including diligent ones) that mistake items of a class i for items of a class j (for example, confuse pictures of actresses Ellen Barkin and Cameron Diaz, or Pancakes with Russian *Blini*) often because they are similar or because the task description has failed to explain the differences. This problem can lead to a subset of workers consistently making errors, even on tasks that are apparently easy. For example, in a Google image

Figure 6.1: Examples of common confusions in flags and food as well as actors - in this case intentionally trying to be similar to a person.

search for "Monaco flag", 3 of the top 6 results (50%!) are wrong - and indeed, confused with the flag of Indonesia, and, to a lesser degree, Poland. We get similar error rates in many cases (e. g., Yams and Sweet potatoes, see Figure 6.1), pointing to the fact that for some classes of objects it is very likely to get erroneous ground truth labels, and that sources one could consider reputable (such as flags shop) contribute to the confusion. As we will show, *confusion can also occur in cases where the overall workers accuracy is very high* and therefore likely to go unnoticed since it is not a problem that surfaces across the board, but for some classes only. If left undetected and uncorrected, this leads to consistently wrong labeling for some classes, including the most popular and widely used ones, from majority voting (MV) to Dawid-Skene (DS).

In this chapter, we define the problem and propose an algorithm for detecting such confusions. *Our primary goal is to detect confusion early in the crowdsourcing process* (that is, after a small number of votes), so that we can alert the task designers and improve the task. Our *secondary goal* is to be able to pre-process crowdsourced labels to detect and *correct confused labels*. Once we do this we can then apply our favorite vote aggregation method to get class labels, from MV to DS to the many interesting variations of Expectation Maximization proposed in the literature.

Specifically, we approach the problem by modeling confusion of observations as random variables by introducing the concept of clusters of confused items, i. e., groups of items which are likely to be confused by the crowd. We derive conditional probability functions

for item values as well as the accuracy of workers (sources) and we show that the existing inference procedures (like expectation maximization) cannot be efficiently applied to our case. To address that, we propose a novel inference algorithm based on sampling using Markov Chain Monte Carlo (MCMC).

We evaluate the proposed algorithm over both synthetic datasets and crowdsourcing experiments, aiming at collecting datasets of different nature and difficulty. We show how the proposed inference procedure can be integrated with the main state-of-the-art aggregation techniques, that it scales both in the number of items and in the number of crowd workers, and that it significantly outperforms commonly used aggregation methods especially when the number of votes per item is relatively small (in the single digit range), as it is common in many crowdsourcing tasks. We also analyze in which cases strong baselines such as DS become competitive even in the presence of confused labels. We show that the accuracy of the proposed algorithm in *detecting* confusions is high (up to 98%), while accuracy that results from *correcting* confusion errors varies with the dataset and vote aggregation method chosen, but in general is not as high. To cope with this issue we then propose a simple but effective automated method for modifying the crowdsourcing task once confusion is detected to reduce the probability of further confusion occurring again as the crowdsourcing task proceeds.

6.1 Background

6.1.1 Truth Discovery

The problem of *truth discovery*, i. e., integrating data from sources which provide conflicting data, has been extensively studied in the last decade. (In this work sources(of information) and workers are interchangeable.) A number of approaches which model the accuracy of sources and probability of facts, termed *truth finders*, have been proposed [Li et al., 2016]. Typically, truth finders take the information about observed item labels and output the accuracy of sources as well as the probability of item labels being true. There are several methods truth finders employ such as link based analysis [Kleinberg, 1999; Brin and Page, 1998] (to identify authoritative sources), optimization based methods [Aydin et al., 2014; Li et al., 2014a] (formulate the truth discovery as an optimization problem), and finally, probabilistic graphical models [Pasternack and Roth, 2011; Wang et al., 2015] (based on variables and their dependencies). Moreover, modern truth finders also take into account various factors such as dependency between sources [Dong et al., 2009], heterogeneity of values [Li et al., 2014b], data correlation [Pochampally et al., 2014], different types of source accuracy [Zhao et al., 2012] and so on.

Truth discovery with confusion errors is a challenging problem both from the efficiency

and effectiveness standpoints. First, we need to extend the existing truth finders with the ability to model and reason about possible confusion errors. No existing truth finder solution is capable of doing that and it is not trivial because we need to specify all probabilistic dependencies between sources and items. Second, in order to detect and take into account confusion errors we need to test all possible confusions and see which of them are likely to exist. That is a computationally expensive operation which requires a search over an exponential number of possibilities (there are 2^{N_c} states for N_c possible confused observations).

Crowdsourced classification has also been studied within the Human Computation community, following the seminal work of [Dawid and Skene, 1979]. These studies cover a diversity of probabilistic data models, estimate the labeling procedure in the presence of items of different difficulties [Whitehill et al., 2009; Yang et al., 2018], consider the confidence of workers' answers [Satoshi Oyama, 2013], apply EM-based algorithms to labeling in the presence of confusion matrix [Liu and Wang, 2012], discover truth of spatial events in mobile crowdsourcing [Robin Ouyang, 2016]. A survey [Jing Zhang, 2016] investigates the models and algorithms on truth discovery inference in crowdsourcing as well as summarizes existing datasets and available tools [Robin Ouyang, 2016]. Recently, guidelines and practices for implementation and task design have been proposed for high-quality label collection via crowd and AI collaboration [Alonso and Marchionini, 2019; Amershi et al., 2019].

6.1.2 Confusion of labels

The previous work in truth discovery falls short of finding and repairing such confusions. A typical truth finder or crowd classification algorithm would treat a confused labeling as workers' errors and lower their accuracy estimate in general, as opposed to recognizing that workers are generally accurate but on specific sets of labels. To the best of our knowledge only few papers deal with the problem, besides the above-mentioned Dawid-Skene on which we come back in the experiments section as it is part of our baselines. [Liu and Wang, 2012] study confusions and propose an efficient way to derive a confusion matrix for each worker. Their approach is tailored to situations where the number of workers is small (they focus on highly trained and skilled workers who rate a large number of items and on cases where the item-worker matrix is dense, testing the approach with three workers). [Giancola et al., 2018] also focus on workers' confusion matrix and propose the use of worker *style* matrices for every worker aiming at efficiently aggregating labels that can be systematically confused by a worker in text clustering and image segmentation tasks. The approach is focused on performing permutation-invariant inference of class labels, in the sense that it is robust to worker-specific class permutations (e. g., different workers

assigning different names to the class labels in a clustering problem). Our problem is different in that we do not focus on permutations, but on identifying confusions in labeling where class labels are given and (from the perspective of the task designer) clear, such as labeling picture with flags or food or names of famous actors, based on whatever the task requires. We also do not aim at learning parameters for workers' confusions (that is, at learning parameters for each worker) as our interest is instead in *groups of classes* that can be confused. Therefore the number and types of parameters of our model that have to be learned from data are chosen both to do this effectively and to be able to scale in the number of workers, items, and labels.

The topic of confusion is also gaining popularity in the context of machine learning, to identify and diagnose weaknesses of a ML model which often occurs in the presence of easily "confusable" classes and in general of noisy training data. For example, [Jin et al., 2017a] propose a method to detect classes with high probability of confusion based on the output of an image classification ConvNet. The authors observe that the problem often arises due to a combination of weaknesses of the classifier and errors in the training data: Indeed, errors in the training dataset are known to significantly hamper the performance of the trained model [Russakovsky et al., 2015], and the risk of noisy crowdsourced labels increases with the complexity of the problem and with the granularity of the classes [Van Horn et al., 2015]. The solution proposed is essentially based on identifying cliques of classes such that, for many items, classes in the cliques are among the most likely in the opinion of the ML model. Our problem is different as in crowdsourcing we have a crowd of votes (without explicit confidence information) rather than one voter (the ML model) that provides a distribution of class probabilities for each item.

6.2 Model and Approach

6.2.1 Modeling Confusions as First-Class Citizens

We consider classification tasks where we have as input a set (I, S, L) where $I = \{o_1, \ldots, o_m\}$ is the set of items to classify, S is a set of sources of information (in crowdsourcing, these are crowd workers), and $L = \{L_1, .., L_m\}$ is a set of all possible labels on all items. Each item o_i have a number of possible labels (classes) we can assign $L_i = \{l_i^1, \ldots, l_i^{k_i}\}$ where k_i is the total number of distinct labels about o_i. Table 6.1 contains a summary of the notations used in this chapter.

The votes of workers are denoted as $\Psi = \{\psi_{s,l}^i\}$, where $\psi_{s,l}^i = 1$ if source s labeled item o_i with label l, and $\psi_{s,l}^i = 0$ otherwise. Therefore, once we collect the votes we have a dataset $D = \langle I, S, \Psi, L \rangle$.

We extend this basic model by explicitly modeling confusions. Specifically, we add the

Sign	**Specification**
$S = \{s_1, \ldots, s_n\}$	set of workers (sources)
$I = \{o_1, \ldots, o_m\}$	set of items
$L = \{L_1, \ldots, L_m\}$	set of all labels, L_i are labels of o_i
o_i^*	predicted label for item o_i
l_i^*	ground truth label of item o_i
$\Psi = \{\psi_{s,l}^i\}$	set of all votes
Ψ_k	observations on the items in C_k
Ψ^s	observations of worker s
$\hat{\Psi}^i$	observations on o_i including confused ones
$C = \{C_1, \ldots, C_{n_k}\}$	set of all clusters
C_k^{-i}	set of items in C_k except o_i
A_s	accuracy of worker s
G_k	confusions of cluster C_k
$G_{i,s}$	confusion of observation ψ_s^i
$G_k^{-i,s}$	confusions in cluster k except $G_{i,s}$
π_k	probability of confusion in C_k
α, β, γ	hyper-parameter

Table 6.1: The summary of notations used in the chapter.

notion of *clusters* where all items are split into a set of mutually exclusive clusters, denoted by $C = \{C_1, \ldots, C_{n_k}\}$ where $C_k = \{o_i, o_j\}$ is a group of items whose labels we believe can be confused, such as in the example of Figure 6.1 where pictures of "monaco", "indonesia" can be in a cluster. We come back later as to how such clusters can be identified, for now we assume they are given.

6.2.2 Truth Finder with Confused Observations

Given this model, and a dataset, we aim at determining confusions in the dataset. We base our work on truth finder approaches. A truth finder $\mathcal{F}$ is a function that takes the dataset D as input, and outputs a set A of accuracy estimations (one per worker) and the class probability distribution P for each item. $\mathcal{F} : D \rightarrow \langle P, A \rangle$ where for each item $o_i \in I$, $P(o_i^* = l) \in [0, 1]$ is the probability that item o_i has label l (we use the terms item and

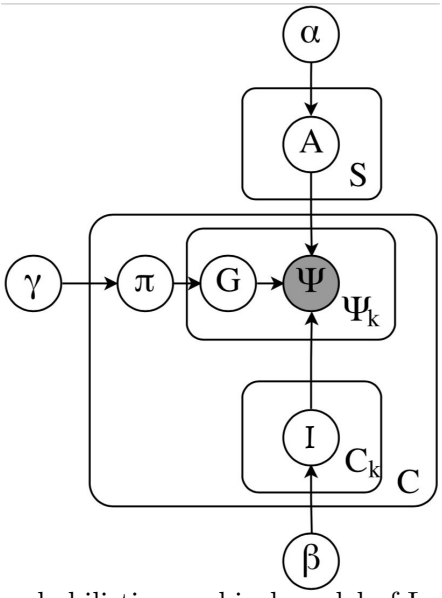


Figure 6.2: The probabilistic graphical model of Latent Truth Finder.

object interchangeably in the following).

The typical truth finder goal is to infer the unobserved variables (the true labels) given the observable variables (the crowd votes). We extend the basic Bayesian truth finder with clusters and define its generative model based on the plate model shown in Figure 6.2. In the figure, nodes indicate random variables, parameters, and hyper-parameters, and dark-shaded node denote observations made by workers. G, I are latent (or non observed) variables; A, π are the parameters; α, β, γ are hyper-parameters.

For each worker we draw its accuracy from a Beta distribution: $A_s \sim Beta(\alpha_0, \alpha_1)$, where α_0 and α_1 represent our prior belief about the accuracy of workers. α_0 can be seen as the number of times a worker gave the correct label whereas α_1 is the number of incorrect labels.

For each cluster $C_k \in C$, we draw its probability of confusion labels, denoted by π_k, from a Beta distribution: $\pi_k \sim Beta(\gamma_0, \gamma_1)$. Similarly to A_s, γ_0 and γ_1 represent our prior belief about how many times workers confused labels within that given cluster in the past.

For each crowd vote ψ_s^i in cluster C_k, i. e., $\psi_s^i \in \Psi_k$ and $\psi_s^i \in L_i$, we draw a confusion binary label, denoted by $G_{i,s}$, from a Bernoulli distribution: $G_{i,s} \sim Bernoulli(\pi_k)$, where $G_{i,s}$ is 1 if worker s confused label on item o_i, and 0 otherwise. π_k is a probability of confusing labels when reporting observations on cluster C_k. Notice that in the model we do not consider the *direction* of the confusion, that is, whether in cluster C_k the confusion is symmetric or not symmetric, for example, workers confuse l_1 with l_2 but not l_2 with l_1. As we will see, once confusion is detected by the inference procedure, then identifying

direction can be easily done.

Finally, we draw a true value for an item o_i from a multinomial distribution with β_i Dirichlet prior: $o_i^* \sim multi(\beta_i)$ β_i is a $|L_i|$-dimensional parameter which specifies the multinomial prior for item o_i.

6.3 Inference

Based on the generative model presented in Section 6.2 we propose an inference algorithm to detects confused observations. Once we have detected them, we can *correct* them (the algorithm helps in finding the probabilities of true labels) or *prevent* them, by modifying the crowdsourcing task as discussed later. Our inference task is *maximum a posteriori* (MAP) where we look for an assignment of model parameters and hidden variables that maximizes the likelihood of the observed data. Given our model, the likelihood is defined as follows (we denote it with LK to distinguish it from labels):

$$LK(A, \pi : \Psi) = \prod_{C_k \in C} \sum_{I,G} \prod_{\psi_s^i \in \Psi_k} P(\psi_s^i \mid A, \pi, I, G; \alpha, \beta, \gamma) \tag{6.1}$$

In the above formula, for each cluster C_k we compute a product of the probabilities of observed data (Ψ_k) marginalized by the hidden variables, I and G. Note that we will omit the hyper parameters α, β and γ in the following formulas.

Our task is to find $\hat{A}$ and $\hat{\pi}$ such that the likelihood function is maximized: $\hat{A}, \hat{\pi} = \underset{A,\pi}{argmax}\ LK(A, \pi : \Psi)$. Once we found $\hat{A}$ and $\hat{\pi}$ we can easily compute the values of hidden variables, i. e., I and G. We propose to employ an approximation inference procedure based on Markov Chain Monte Carlo which samples from the joint probability distribution and use those samples to estimate the hidden variables and parameters. We refer to this algorithm *MCMC-C* (the extra C is for confusion). In order to use MCMC we need to specify all conditional probability distributions (CPDs) for each variable of our model. Below we present the CPDs we use.

First, we define a CPD for items $o_i \in I$. According to our data model, we sample a true label for item o_i using a multinomial distribution with the following probabilities:

$$P(o_i^* = l_i^* \mid \Psi_k, C_k^{-i}, A, G_k^{-i,s}, A, \pi_k) \propto \beta_{o_i} \prod_{\psi_s^j \in \hat{\Psi}^i} A_s^{\delta(\psi_s^j, o_i^*)} (1 - A_s)^{1 - \delta(\psi_s^j, o_i^*)}$$

where $\delta(i,j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$ is the Kronecker delta function and $\hat{\Psi}^i$ is a set of observations on item o_i including those which were confused (based on the current assignments of G), C_k^{-i} is a set of items of cluster C_k except item o_i and $G_k^{-i,s}$ is a set of confusions G_k except

confusion $G_{i,s}$, Intuitively, we sample a true value for o_i based on our current knowledge of worker accuracy and observation confusions.

Similarly, we sample confusion variables from a Bernoulli distribution using the following probabilities:

$$P(G_{i,s} = 0 \mid \psi_s^i, C_k, A_s, \pi_k) \propto (1 - \pi_k) A_s^{\delta(\psi_s^i, o_i^*)} (\frac{1 - A_s}{|L_i|})^{1-\delta(\psi_s^i, o_i^*)} \tag{6.2}$$

$$P(G_{i,s} = 1 \mid \psi_s^i, C_k, A_s, \pi_k) \propto \frac{\pi_k}{|C_k| - 1} (\sum_{o_j \in C_k, i \neq j} A_s^{\delta(\psi_s^i, o_j^*)} (\frac{1 - A_s}{| \bigcup_{o_j \in C_k} L_i \mid -1})^{1-\delta(\psi_s^i, o_j^*)})$$

The probability that there is no confusion, $P(G_{i,s} = 0)$, depends on the cluster no confusion probability $1 - \pi_k$ as well as the worker accuracy A_s if the item predicted true label o_i^* the same as the vote ψ_s^i or $\frac{1-A_s}{|L_i|}$ otherwise (assuming that each false label is equally likely). In the case of $P(G_{i,s} = 1)$ we take into account all possible confusions within the cluster C_k with equal prior probabilities $\frac{1}{|C_k|-1}$ and we add A_s if the the vote ψ_s^i coincides with o_j^* or $\frac{1-A_s}{|\bigcup_{o_i \in C_k} L_i|-1}$ otherwise. In the latter case we assume that if there is a confusion than the space of false labels consists of all unique votes within a cluster minus 1 (true label). We sample the accuracy of workers from a Beta distribution with the following parameters:

$$A_s \sim Beta(\alpha_0 + N_+^s, \alpha_1 + N_-^s) \tag{6.3}$$

where

$$N_+^s = \sum_{\psi_s^i \in \Psi^s} (1 - G_{i,s}) \delta(\psi_s^i, o_i^*) + \frac{G_{i,s}}{|C_k| - 1} \sum_{o_j \in C_k^{-i}} \delta(\psi_s^i, o_j^*) \tag{6.4}$$

$$N_-^s = \sum_{\psi_s^i \in \Psi^s} (1 - G_{i,s})(1 - \delta(\psi_s^i, o_i^*)) + \frac{G_{i,s}}{|C_k| - 1} \sum_{o_j \in C_k^{-i}} (1 - \delta(\psi_s^i, o_j^*)) \tag{6.5}$$

Ψ^s is a set of votes given by worker s. N_+^s is the number of times a worker s gave the same votes as the corresponding item true label. In the case of a confusion we assume that it measures any of the other $|C_k| - 1$ items at the same probability. N_-^s is the count of worker s observations when it reported a false label (with the similar way of counting confusions).

Finally, we sample the cluster confusions from a Beta distributions using the counts of confused and not confused labels (i. e., using $G_{i,s}$):

$$\pi_k \sim Beta(\gamma_0 + \sum_{G_{i,s} \in G_k} G_{i,s}, \gamma_1 + \sum_{G_{i,s} \in G_k} (1 - G_{i,s})) \tag{6.6}$$

Given all CPDs presented above the MCMC inference algorithm is described in Algorithm 5.

Algorithm 5: **MCMC Inference Algorithm**

Input: Ψ, α, β, γ
Output:$\hat{P}(O)$
Initialize A_s, $G_{i,s}$, o_i^* and π_k by drawing from the priors

(1)
foreach $A_s \in A$
(2) $N_+^s \leftarrow \sum_{\psi_s^i \in \Psi^s} (1-G_{i,s})\delta(\psi_s^i, o_i^*) + \frac{G_{i,s}}{|C_k|-1} \sum_{o_j \in C_k^{-i}} \delta(\psi_s^i, o_j^*)$
(3) $N_-^s \leftarrow \sum_{\psi_s^i \in \Psi^s} (1-G_{i,s})(1-\delta(\psi_s^i, o_i^*)) + \frac{G_{i,s}}{|C_k|-1} \sum_{o_j \in C_k^{-i}} (1-\delta(\psi_s^i, o_j^*))$
(4) **foreach** k **to** *iter_num*
(5) **foreach** $o_i \in O$
(6) draw $o_i^* \sim multi(\prod_{\psi_j^s \in \hat{\Psi}^i} A_s^{\delta(\psi_s^j, o_i^*)}(1-A_s)^{1-\delta(\psi_s^j, o_i^*)})$
(7) $updateN_+(o_i^*)$
(8) $updateN_-(o_i^*)$
(9) **foreach** $G_{i,s} \in G$
(10) draw $G_{i,s} \sim \pi_k A_s^{\delta(\psi_s^i, o_i^*)}(1-A_s)^{1-\delta(\psi_s^i, o_i^*)}$
(11) $updateN_+(G_{i,s})$
(12) $updateN_-(G_{i,s})$
(13) **foreach** $A_s \in A$
(14) draw $A_s \sim Beta(\alpha_0 + N_+^s, \alpha_1 + N_-^s)$
(15) **foreach** $\pi_k \in \pi$
(16) draw $\pi_k \sim Beta(\gamma_0 + \sum_{G_{i,s} \in G^k} G_{i,s}, \gamma_1 + \sum_{G_{i,s} \in G^k} (1-G_{i,s}))$
(17) **return** $\hat{P}(O)$

Notice that the complexity of Algorithm 5 is proportional to the number of iterations, observations and items in clusters: $O(itern * |\Psi| * max(|C_k|))$ The Algorithm 5 computation overhead (with respect to non confusion aware methods) comes from the necessity to visit all the items within a cluster when we compute counts for A_s and o_i, i. e., for each item, we take into account the observations which were switched *from* and *to* it. However, in practice the $max(|C_k|))$ doesn't affect the overall computational cost a lot since there are not many items within a cluster and totally not all items are involved into clusters.

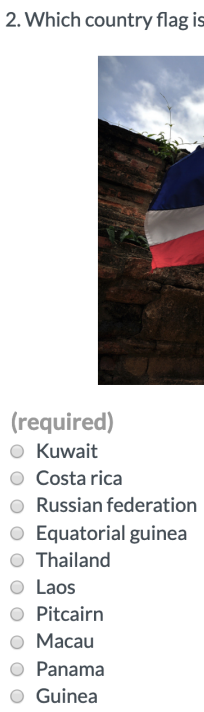


Figure 6.3: Crowdsourcing task to collect labels on flags.

6.4 Experiments and Confusion Prevention

We run experiments with the goal of assessing the algorithms in terms of its ability to i) *detect* confusion and ii) *correct* them. We then show a simple but effective way to semi-automatically *prevent* confusions, once detected.

6.4.1 Datasets

We evaluate the model and algorithm first with a synthetic dataset, to assess performances as we vary the characteristics of the data, and then via four crowdsourcing experiments of different nature and difficulty. The synthetic dataset is obtained via a custom synthetic data generator that allows us to vary the number of items, sources, labels (classes), the average accuracy of sources, the probability of confusion for the clusters as well as other parameters.

We then run four crowdsourcing experiments asking workers to label photos of flags, faces of celebrities, food images, and to assign the title of a movie given the plot. The first three datasets are based on images while the fourth is on text labeling. The tasks are analogous to the one shown in Figure 6.3: we show an item and a set of possible labels, which we design to include, among many others, the true one as well as the label for a similar one where confusions are possible, if any. All tasks where run on the *FigureEight*

platform. Table 6.2 shows task and data statistics such as number of items, workers, votes. Overall we collected more than 5100 votes in the tasks. As an example, for the flags dataset we included photos of flags of 60 countries and collected 16 votes per image on average. 220 distinct workers took part in the flags task, with an overall accuracy of 90%. We then selected initial hypotheses of pairs of labels (clusters) that we thought could be source of confusion, such as *Moldova* and *Romania*. For flags we selected 13 such clusters, and the errors we define to be of confusions (that is, between items that are in the same cluster) are 7.8% and involve 10 out of the 13 clusters (notice that nearly all errors are confusions). Some confusions are one-directed (e. g., Moldovan flags labeled as Romanian) and others bi-directional (Ireland and Ivory coast labels are swapped).

Task Property	**Flags**	**Faces**	**Food**	**Plots**
# of classes	81	72	45	111
# of workers	220	21	177	122
# of items	100	48	76	100
# of votes	1600	349	1220	1937
votes per item (avg ± std)	16 ± 5	7 ± 3	16 ± 10	19 ± 1
votes per worker (avg ± std)	7 ± 5	16 ± 6	7 ± 4	16 ± 12
% of confused votes	7.8%	19.2%	23.8%	6.1%
workers' accuracy	90%	70.2%	68.8%	90.2%

Table 6.2: The flags, faces, food, and plots datasets statistics.

6.4.2 Confusion Detection

We start by testing the ability of the algorithm to detect confusions, our primary goal. Specifically, we are interested in the average probability of confusion detection, that we call *G-Accuracy* (G denotes confusions). It is computed as follows:

$$G\text{-}Accuracy = \frac{\sum_{G_{i,s} \in G} \hat{P}(G_{i,s} = G^*_{i,s})}{\mid G \mid} \tag{6.7}$$

where $\hat{P}(G_{i,s} = G^*_{i,s})$ is the estimated probability of identification a correct value ($G^*_{i,s}$) for a confusion variable ($G_{i,s}$), i. e., whether or not a confusion appeared, and $\mid G \mid$ normalizes

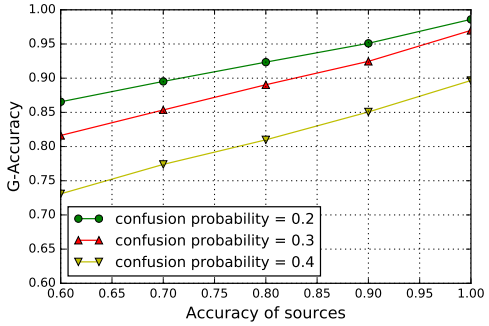


Figure 6.4: G-Accuracy of `MCMC-C` for the varying sources accuracy.

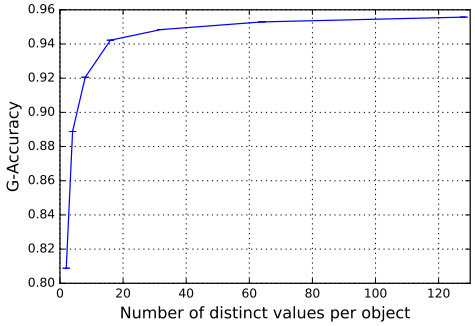


Figure 6.5: G-Accuracy of `MCMC-C` as the number of classes grows.

Confusion detection (all data, see Table 2)

Dataset	G-acc	Precision	Recall
Plots	98.7+-0%	99+-1%	93+-5%
Flags	97.6+-1%	91+-9%	76.8+-8%
Faces	85.7+-1%	78+-5%	59+-5%
Food	82.8+-1%	85+-4%	50+-5%

Confusion detection (5 votes per item)

Plots	97.1+-1	91.6+-12%	84.7+-10
Flags	96+-2%	84.6+-10%	69.1+-10%
Faces	83+-2%	73.3+-8%	55+-6%
Food	84.3+-2%	76.1+-9%	58.6+-8%

Figure 6.6: Results on real-world crowd datasets.

Improvement in p-accuracy (all data, see Table 2)

Dataset	MV original | improved	TruthFinder original | improved	MCMC original | improved	D&S original | improved	MCMC-C
Plots	84 | 96	98 | 99	98 | 99	92 | 96	**99**
Flags	83 | 93	98 | 97	**99 | 97**	71 | 87	97
Faces	58 | 67	79 | 79	79 | 80	42 | 54	**80**
Food	63 | 72	67 | 76	74 | 76	**77 | 79**	76

Improvement in p-accuracy (5 votes per item)

Plots	69 | 86	89 | 92	86 | 92	29 | 76	**92**
Flags	68 | 81	**85 | 85**	80 | 84	57 | 78	83
Faces	56 | 64	**78 | 78**	76 | 76	38 | 53	77
Food	56 | 66	67 | 71	65 | 71	47 | 66	**71**

Figure 6.7: Improvement in p-accuracy after error correction.

the metric by the total number of votes given on items that belong to a cluster, so that it is in the $[0, 1]$ interval.

Figure 6.4 shows the results of applying MCMC-C while varying the accuracy of workers A and probability of confusion π. Our default parameters are 30 workers with average accuracy 0.9, 5000 items, and 50 classes. As expected the output accuracy grows linearly with the worker accuracy. Moreover, the higher A and reasonably low π the MCMC-C gives more accurate results, particularly, for $A = 1$ and $\pi = 0.2$ the accuracy of identifying confusions is 98%. Figure 6.5 show instead that accuracy grows with the number of possible classes (here shown keeping $\pi = 0.2$), as it becomes easier to separate confusions. On crowdsourcing experiments (Figure 6.6), confusion detection accuracy is also fairly high especially in terms of precision. Numbers are lower for food as the overall accuracy is lower for that case.

These results are based on a full dataset, which includes many votes per item (recall Table 6.2), a number much higher than typical crowdsourcing tasks. Figure 6.6 also shows results considering only 5 randomly selected votes per item - and we discuss later how performances vary with the number of votes per item.

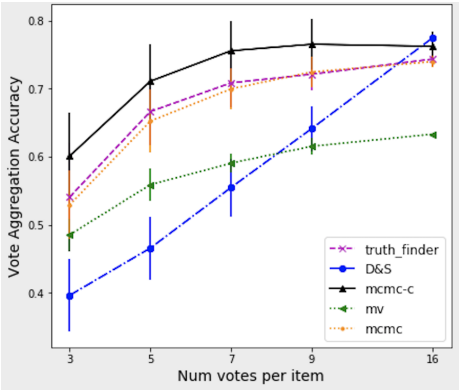


Figure 6.8: Aggregation accuracy as the number of votes per items change - food.

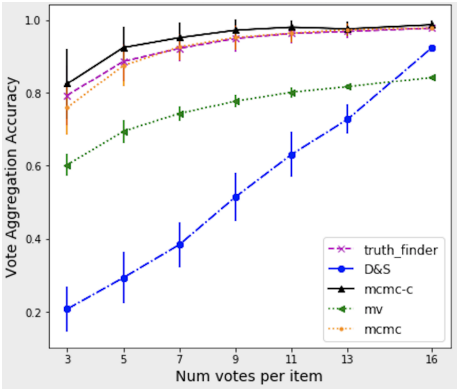


Figure 6.9: Aggregation accuracy as the number of votes per items change - plots.

6.4.3 Confusion Correction

Our secondary goal is to leverage MCMC-C to improve classification accuracy once votes have been collected. MCMC-C attempts at identifying which votes in a cluster are confused, therefore enabling error correction (reassigning confused votes to correct items), at least in the case of clusters of two labels discussed here. The impact of error correction on the overall classification accuracy also depends on the classification algorithm adopted. We therefore experiments with a set of state of the art vote aggregation algorithms and test the effect on adding confusion correction on top. So that first we correct confusions and the run vote aggregation algorithms. Specifically, we experiment with *Majority Voting*[1] (`mv`), *Dawid-Skene* (D&S), and the already mentioned *Expectation Maximization* (`em`)-based truth finder, as well as with a base MCMC version such as *MCMC Sampling Algorithm* (`mcmc`) [Zhao et al., 2012] (Effect with other algorithms, such as SUMS [Kleinberg, 1999], Investment, PooledInvestment, Average-Log [Jeff Pasternack, 2010], is described in the supplementary material).

To assess the improvements we measure the accuracy as the probability assigned by the algorithm to the true class, and average across all items. That is, we measure $p\text{-}Accuracy = \underset{o_i \in I}{\text{mean}}(\hat{P}(o_i^* = l_i^*))$, where $\hat{P}(o_i^* = l_i^*)$ is the estimated probability of item o_i having ground truth label l_i^*, and compare results before and after correcting for what MCMC-C believes to be confusion errors.

Figure 6.7 shows the improvement in p-accuracy for the crowdsourced datasets. The top part of the figure shows that applying the correction has very large effects in terms of improving MV and D&S in many cases, while the effect on TruthFinder is small. Again, the bottom part of the figure shows results where we only collect 5 votes per item, while Figures 6.8 and 6.9 show how performances vary with the number of votes per items for

[1] For mv, we take as probability of class label for an item the proportion of votes with that label.

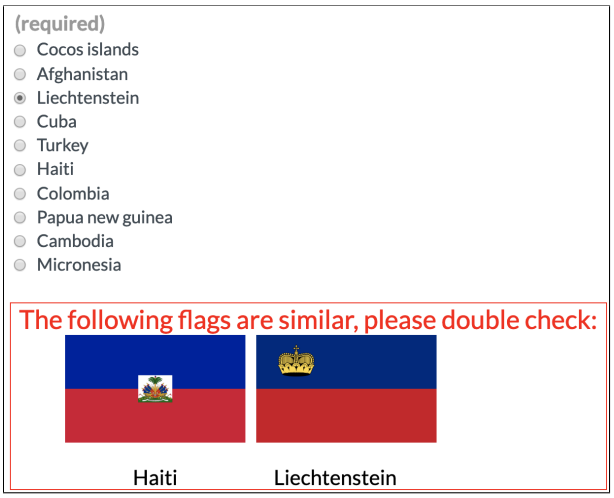


Figure 6.10: Confusion prevention crowd task.

the datasets where D&S performs *best* in the full dataset. The reason why D&S does not give good results with low numbers of votes per item is that in these datasets we have rather large number of classes (from 45 in *food* to 111 in *plots*), and in this case confusion matrices have a large number of cells (over 10K for *plots*), so that constructing such matrices (per worker) accurately requires more data. In this case, as the number of data points increases, D&S performs well in the *plots* dataset despite the large number of classes also because the workers' accuracy is high and the confusion is low, the lowest in our datasets. In general, we observe that with number of votes that are in line with typical scenarios, correcting for detected confusions before aggregating results in at least the same accuracy, and very often a much improved accuracy with respect to performing aggregation without correction.

6.4.4 Confusion Prevention

A main reason to detect confusion early is the process is the possibility of preventing it. MCMC-C can detect confusions even with a small number of votes per item (see charts in supplementary material). Once confusion is detected between a pair of labels, a simple and automated prevention mechanism is to modify the crowdsourcing task by pointing out the possibility of confusion every time the worker selects a label in this pair.

We show an example of this in Figure 6.10, where once the flag of Haiti or Liechtenstein is selected, a popup automatically appears showing the similarity (and therefore the possible confusion) and asking for verification. We run crowdsourcing experiments over flags, plots and food experiments with the prevention mechanism and reduction in confused labels are from 6.1% to 2% for plots, 23.8% to 19.3% for food, 7.8% to 2.6% for flags.

6.4.5 Conclusion

We have shown that MCMC-C is an effective to detect confusion over a variety of datasets, and that once confusion is detected, it can then be prevented to a large extent with simple modifications to the task. The algorithm scales well to large datasets (both in items and workers).

Part of our work included a comparison with D&S as a baseline, and of what happens when we preprocess confusions and then apply D&S. We are specifically interested in D&S as a main representative of algorithms that do aim at identifying confusion matrices and therefore in a way do deal with confusions. However, D&S per se does not output classes that some workers may confuse, but it "simply" produces estimated labels and per-worker confusion matrices. In principle it is possible to use such matrices to determine pairs (or cliques) of classes that can be source of frequent confusions, and this requires a way to map the set of worker-specific matrices into sets of possibly confused labels. However, how to do this is not obvious and requires further research. Simply taking matrix cells off the diagonal where the average value is high (denoting that workers frequently mistakes the estimated correct label - in the row - with the voted label - in the column) often leads to poor results especially in the conditions discussed above (large number of cells and relatively sparse matrices with workers votes). If workers' accuracy is also not very high, then D&S has a hard time distinguishing errors and confusions.

To show how D&S behaves in the presence of classes that some workers confuse, we plot in Figure 6.11 three *confusion matrices histograms* obtained by D&S, corresponding to three (simulated) experimental conditions. Each matrix cell contains an histogram plotting the number of workers (y axes) that have a specific confusion value for that cell (the X axis is divided in buckets of size 0.1). So for example, cell [1, 2] (first row, second column) shows a histogram denoting how many workers have a given confusion value for that cell. The expectation is that cells on diagonal have bars close to 1, and off-diagonal cells have most of the mass towards zero. All matrices correspond to experiments with 5 classes, workers' accuracy in the [0.7-0.9] range, and have a varying percentage of workers that confuse class 1 and 2. Therefore, we assume to observe signals for confusions in cells [1, 2] and [2, 1]. The dataset for the top matrix has 50% of workers confused, and has 5 votes per item, 25 votes per worker. The one in the middle also has 50% of workers confused, and 3 votes per item, while the bottom one has 5 votes per item but only 10% of workers are confused between classes 1 and 2. What we can see is that while for the top matrix we could infer confusions, for example by observing that the histogram has mass close to both extreme and not just one (see in particular cell [1, 2]), when the number of votes per items goes down (matrix in the center) or the percentage of confused workers is a probably more realistic 10%, making confusion determination becomes challenging.

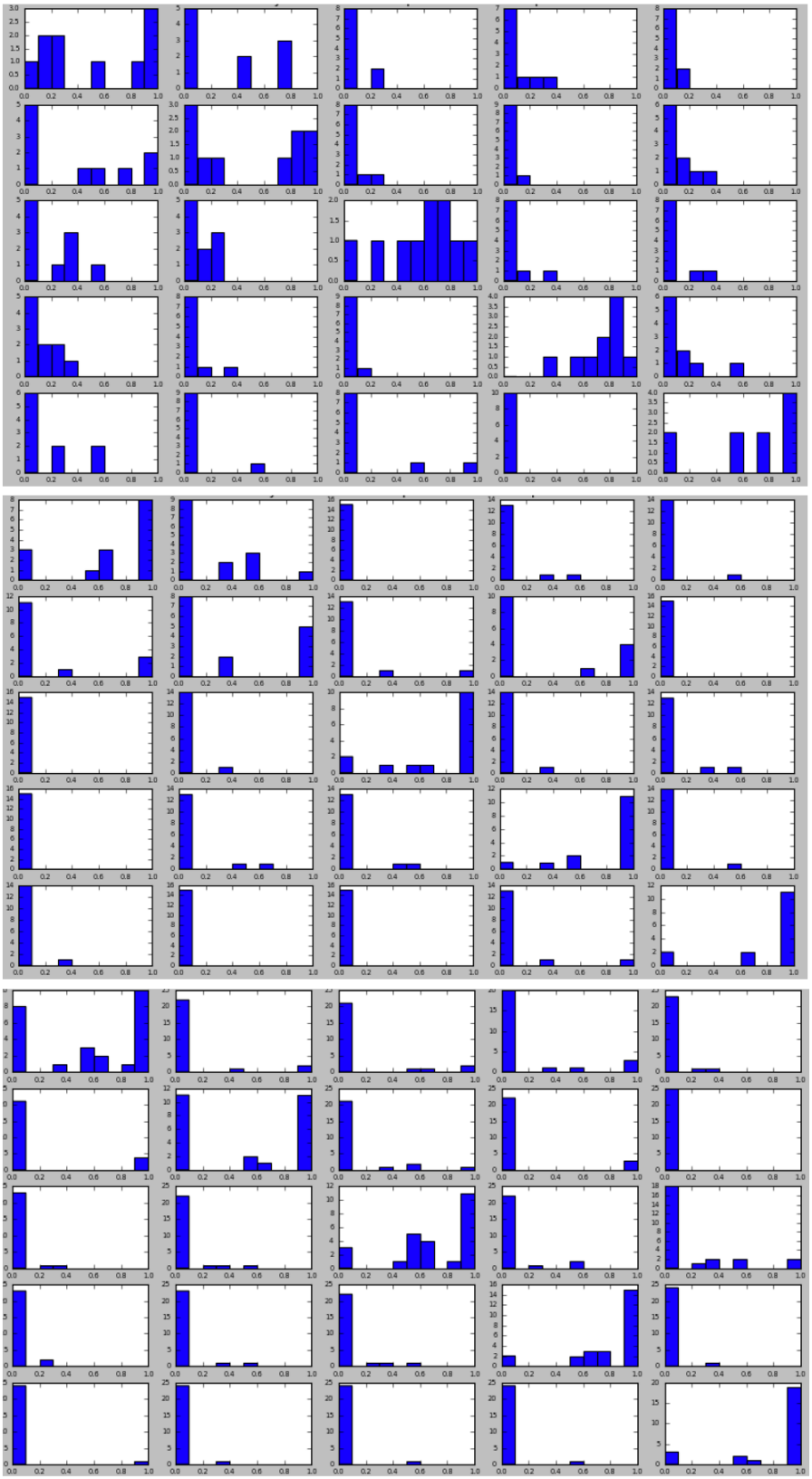


Figure 6.11: Dawid-Skene for Confusion Detection. Confusion matrices from Dawid-Skene where the Y-axis represents the number of workers and the X-axis represents the confidence value.

Chapter 7

Graph Neural Networks for Entity Matching

Although knowledge graphs (KGs) and ontologies have been exploited effectively for data integration [Pershina et al., 2015; Trivedi et al., 2018; Azmy et al., 2019], entity matching involving structured and unstructured sources has usually been performed by treating records such as arrays or text without explicitly taking into account the natural graph representation of structured sources and the potential graph representations of unstructured data [Getoor and Machanavajjhala, 2012; Gottapu et al., 2016; Mudgal et al., 2018; Dong and Rekatsinas, 2018; Gschwind et al., 2019]. Implicit graph representations of database records have so far been exploited for record-linkage tasks by taking into account the similarity of the attributes between a query and a candidate record. Similarities between records are usually computed by means of multicriteria scoring without considering the direct attribute for information flow [Gschwind et al., 2019]. Additionally, many contextual fields, that are often represented by record attributes, are considered to be uninformative and hence are dropped in preprocessing steps, even though they might contain important structural information about the entities represented by the records.

To address this issue, this chapter introduces a methodology for leveraging graph-structured information in entity matching. The main idea consists of exploiting KGs to create distributed representations of the nodes, so that entities related to each other (e. g., having the same entity type) in the KGs are closer in the embedding space. KGs can be derived from the database at hand, or they can be taken from external sources, similarly to distant-supervision approaches. For example, the database in our deployment contains information about companies and their relations (e. g., in terms of ownership and subsidiaries). Databases that store information about businesses (such as those provided by [Dun & Bradstreet Inc]) explicitly or implicitly incorporate relations between companies

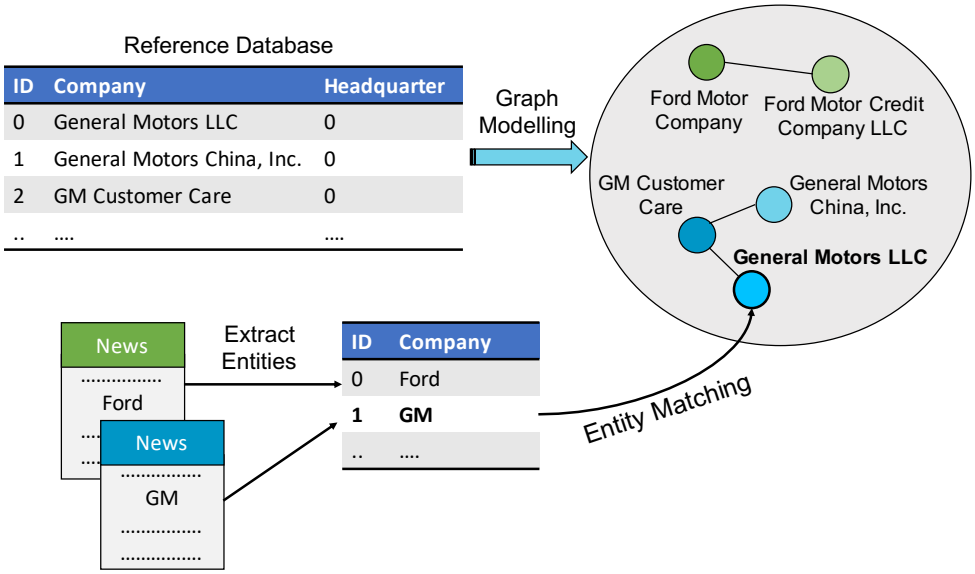


Figure 7.1: Example of entity matching and graph modeling.

and their attributes. This means that companies and their connections can be represented as a graph and stored accordingly. As a simple example, Figure 7.1 depicts graph modeling from a reference database, and shows how company entities, extracted from news articles, are linked to corresponding records.

Using graph representations for both structured and unstructured data sources makes it possible to leverage graph neural networks, thereby allowing all the available context to be used in order to propagate useful information from one node to another. By context we mean all the attributes, connected records and sometimes external information that allows to fully understand and disambiguate entities.

The propagation of useful information from a node to its neighbors allows us to build discriminative node embeddings, which are necessary for higher-level tasks. Recently, graph neural networks (GNNs) yielded promising results for modeling hidden representations of graphs. They have already achieved state-of-the-art performance on several tasks, including link prediction, node classification, and graph generation [Hamilton et al., 2017; You et al., 2018b,a; Zhang et al., 2018; Ying et al., 2018; Xu et al., 2018; Velickovic et al., 2018; Xu et al., 2019]. Modeling entity representations through graphs and building their embeddings using GNNs has the potential to greatly improve performance in the field of data integration as well.

In general, KGs can model different kinds of relations (e. g., companies owning other companies, operating in the same domain, or having headquarters in the same city), and some relations can even be implicit (e. g., by considering transitive closure or by taking

into account the distance between nodes in the graph). For simplicity, we focus on a single type of relation, but the same approach can be extended to consider multiple relations among entities.

Our approach is based on combining Siamese and Graph Convolutional Networks (S-GCN) to learn a distance function between the nodes. At training time, the network is optimized to produce small distances for pairs of nodes belonging to the same entity, and large distances for unrelated nodes. Therefore, the trained network can then be used to assess the similarity between a query and any reference node. S-GCNs are especially useful when the task involves a large number of classes, but only a few examples for each label are available.

One of the most important properties of graph modeling and S-GCNs is their ability to catch important information from the data and their structure. It means that the similarity and dissimilarity of objects can be fully defined by their initial properties and connections, so that only a small amount of labeled data is needed to build informative object representations. This property makes graph modeling with S-GCNs extremely suited for relational databases, as they provide important structural information by definition.

The main contributions of this chapter are as follows:

- we investigate how graph modeling and GNNs can be applied to data integration problems;
- we propose a general approach for modeling hidden representations of entities in structured and unstructured databases;
- we devise various architectures of Siamese Graph Convolutional Networks for data-linkage tasks and evaluate the performance of the models for high scale record linkage tasks over business entity databases.

7.1 Background

Although the idea of developing and studying neural networks capable of manipulating and processing graph-structured inputs dates back more than ten years [Gori et al., 2005; Scarselli et al., 2009], these kinds of models have recently been rediscovered and gained an unprecedented popularity for several different tasks and domains [Battaglia et al., 2018]. Such architectures have proven useful for many problems that require reasoning on a set of discrete entities, such as combinatorial optimization [Khalil et al., 2017] and satisfiability [Selsam et al., 2018] problems. More recently, [Kool et al., 2019] successfully applied a model based on the Transformer architecture [Vaswani et al., 2017] to solving the routing problems on graphs, such as the well-known traveling salesman problem (TSP).

Some of the most interesting and promising contributions in this field include message-passing neural networks [Gilmer et al., 2017] and non-local neural networks [Wang et al., 2018]. In addition, the graph networks introduced by [Battaglia et al., 2018] generalize several previous works, thereby providing a fundamental building block for applying deep learning to structured knowledge.

Recently, GNNs have been exploited for different facets of the entity linking tasks. For example, LinkNBed [Trivedi et al., 2018] is a deep relational learning framework for link prediction in multiple KGs. In that work, joint inference is performed over multiple KG structures that capture contextual information for entities and relations. To mitigate the limited amount of training data, they propose using multitask learning by leveraging additional kinds of data in different scenarios, such as unlabeled and negative instances. In contrast, our framework proposes a distance learning approach together with GCNs to overcome the lack of training data in a consistent way.

Another GNN framework, namely GraphUIL, has been introduced for user identity linkage in [Zhang et al., 2019]. GraphUIL ensures that both local and global information forming the user's social graph is propagated in order to jointly learn useful representations to identity linkage. Learning representations for the nodes in a graph proved beneficial for various tasks, such as node classification, clustering and link prediction [Xu et al., 2018; Monti et al., 2017]. GNNs learn node representations through iterative signal propagation between the hidden features of the neighboring nodes. As shown in [Xu et al., 2018], k iterations of the signal propagation process allow reaching the entire subtree of depth k rooted at a given node. This learning process generalizes the Weisfeiler–Lehman graph isomorphism test [Weisfeiler and Lehman, 1968], indicating that both topology and hidden features of the neighborhood are learned simultaneously [Shervashidze et al., 2011].

State-of-the-art methods for entity matching of unstructured data have recently applied various neural network architectures to learn entity representations. For instance, [Ganea and Hofmann, 2017] describes how an attention mechanism over local context windows generates proper embeddings for document-level entity disambiguation. A novel zero-shot entity matching task with previously unseen entities and a limited amount of training data for specialized domains is introduced in [Logeswaran et al., 2019]. The goal of that work is to build a linker that can generalize to unseen specialized entities from unstructured sources with only textual context being available. Best results have been obtained by a model based on BERT [Devlin et al., 2019], which was pre-trained on a specialized domain corpus and then fine-tuned using a small amount of the available training data.

Recently, [Ma et al., 2018] and [Ktena et al., 2017] have combined GCNs and Siamese neural networks to perform the graph-matching task on functional brain networks. These approaches extract graphs from corresponding MRI images and learn how similar two brain

networks are. A recent work [Liu et al., 2018] has proposed an approach for matching long text documents via training Siamese GCNs on their corresponding graph representations. The authors also presented a method for building a graph representation of a document, in which nodes are different concepts (keywords) and edges are interactions between concepts. One of the most closely related works [Shobeir Fakhraei, 2019], though not on graphs, demonstrated the potential for learning embeddings for entity matching. Siamese networks have been employed to create semantic embeddings for protein and chemical names, and the linkage algorithm then simply performed a 1-NN search in the embedding space.

Our work is different from the studies above for the following aspects:

1. we focus on entity linkage rather than graph matching problems;
2. in our problems, query data can be represented by just one node (or several nodes if additional context information is available) rather than a complete network of a brain, or graph of concept interaction between sentences in a large document;
3. we aim at working with thousands of distinct entities rather than at distinguishing disordered and healthy subjects or learning the similarity of a few 3D image classes.

The expressive power of GNNs is analyzed theoretically in [Xu et al., 2019]. The results of this study show that GCNs have limited discriminative power and cannot distinguish some changes in the graph structures. Though a limitation in the general case, this property is beneficial for record linkage tasks. There is usually some discrepancy between the query entity and the reference database, as, due to the limited context availability, the query entity lacks certain links compared to the more complete representation in the reference database. Also, [Kipf and Welling, 2017] experimentally shows that GCNs with two or three layers perform best for the node-classification task, outperforming deeper configurations. This may be partially connected to the results obtained in the late 1980s, called the universal approximation theorem, which showed that a network with a single hidden layer can approximate any continuous function having compact support with arbitrary accuracy as the width of the layer tends to infinity [Cybenko, 1989; Lin and Jegelka, 2018]. Thus, both shallow and wide networks are universal approximators. The approximation properties of deep and narrow networks were studied in [Lin and Jegelka, 2018], which demonstrated that this kind of networks, having a single neuron per hidden layer, are sufficient to provide a universal approximation as the network depth tends to infinity. These approximation guarantees bring additional understanding of the capabilities of GCNs, and deep networks in general, for interpolating complex decision boundaries that appear in data integration tasks.

E	set of entities
R, Q	databases defined over entities in E
$r \in R, q \in Q$	a record in R or Q, respectively
$e_r, e_q \in E$	entities associated with records $r \in R$ and $q \in Q$, respectively
R^*, Q^*	graphs associated with databases R and Q, respectively
$r^* \in R^*, q^* \in Q^*$	nodes associated with entities e_r and e_q, respectively
R^*_r, Q^*_q	sets of the neighboring nodes of $r^* \in R^*$ and $q^* \in Q^*$, respectively
$\Gamma_R, \Gamma_Q \subset \mathbb{R}^M$	sets of embeddings generated from each node $r^* \in R^*$ and $q^* \in Q^*$, respectively
$\gamma_r \in \Gamma_R, \gamma_q \in \Gamma_Q$	embeddings of nodes r^* and q^*, respectively
$\Gamma^k_q \subseteq \Gamma_R$	k nearest neighbors of the embedding γ_q in Γ_R

Table 7.1: Notation.

7.2 Problem Statement

We assume to have two different databases R and Q, defined over a set of entities E, such that each record $r \in R$ and $q \in Q$ can be regarded as an entity mention and uniquely associated to an entity $e \in E$. Each entity represents a distinct real-world object, and we denote with e_r and e_q the entities associated to record $r \in R$ and record $q \in Q$ respectively. The goal of record linkage is to find a function $\mathcal{F}$ that takes as input the databases R and Q, and outputs a set of matches $\mathcal{M} \subset Q \times R$, such that $(q, r) \in \mathcal{M} \iff e_q = e_r$. Assuming we are matching records in Q against records in R, we will henceforth refer to R as the reference database.

In this work, we envision that research in the area of data integration could benefit from following a novel workflow designed to leverage the underlying relations among records and attributes in the database. More specifically, we propose to extract graph representations of the databases and exploit the self-supervision provided by the graph structure to train machine learning (ML) model for data integration, without the need for human-labeled data. To this end, we divide the record linkage problem into two subtasks:

1. graph modeling and
2. GNN design.

The graph modeling stage is about extracting two graphs R^* and Q^* from the databases R and Q. Note that Q is not necessarily produced from structured sources, as the data in Q can come from free text, such as news articles. Given any two records $r \in R$ and $q \in Q$, we denote their corresponding graph nodes as $r^* \in R^*$ and $q^* \in Q^*$, respectively.

A node $r^* \in R^*$ represents a record $r \in R$ and corresponds to a real-world entity $e \in E$. For instance, in Figure 7.1, R is the reference database and Q is a database extracted from news. Graph nodes r^* and q^* are modeled via the attribute *"Company"*, so the record $r_0 \in R$ with $ID = 0$ is represented by the node $r_0^* \in R^*$, that refers to the entity *General Motors LLC*. Edges between the nodes are modeled via the attribute *"Headquarter"*, so r_1^* (*General Motors China, Inc.*) has an edge to r_0^* (*General Motors LLC*). In certain cases, different nodes can refer to the same entity. The edges of the graphs are defined through the relations provided by the initial databases R and Q. Different types of relations determine different strengths of the information flow from one node to another. The "bandwidth" of the edges can be defined a priori or learned automatically during the GNN training phase.

The second stage is to design a neural network that can be trained natively on the graph extracted from the reference database in order to learn a function $\mathcal{N}$, such that:

$$\mathcal{N}(q^*, R^*) = \begin{cases} r^* & \text{if } \exists r^* \in R^* : e_r = e_q \\ \perp & \text{otherwise,} \end{cases}$$

where $\perp$ means that the node q^* could not be matched to any node $r^* \in R^*$. Note that r^* is a single node, but in general $\mathcal{N}(q^*, R^*)$ can produce several matching nodes from R^* representing the same entity $e_r = e_q$. Following the example above, the node $q_1^* \in Q^*$ (*GM*) is matched to r_0^* as they both represent the same entity *General Motors LLC*.

Table **??** summarizes the notation introduced in this section and provides some additional notation that will be covered in the remainder of the chapter.

7.3 Proposed Approach

We approach record linkage as a two-stage process. The first stage consists of obtaining a graph representation R^* of the reference database R. The second stage involves designing and training a GNN on R^* to perform entity-matching tasks on graphs. In the following, we describe our vision of the entity-matching process and propose the S-GCN (Siamese Graph Convolutional Network) model.

This chapter assumes we can obtain a graph representation of the entities of interest. This representation can be extracted in many ways, either by exploiting structures in the database (links between entities, such as foreign keys) or by using an external source. Several methods have also been introduced in the literature for extracting knowledge graphs from both unstructured data [Das et al., 2018; Gangemi et al., 2016] or semi-structured data [Lehmann et al., 2015]. The main contribution of this work lies in the way the graph structure is leveraged and processed to perform data integration.

7.3.1 GNNs for Data Integration

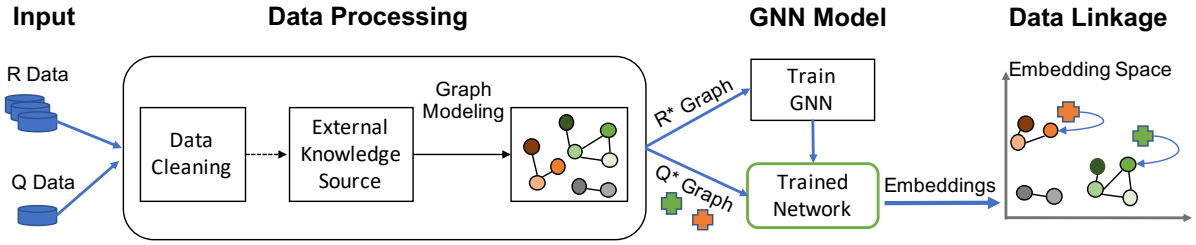


Figure 7.2: Entity matching workflow with graph neural networks.

The reference graph R^* is defined as a directed multigraph, and we denote with $|R^*|$ the number of nodes. We assume to have a feature descriptor $x_r \in \mathbb{R}^D$ for every node $r^* \in R^*$, and we aggregate all these features in a matrix $X_R \in \mathbb{R}^{|R^*| \times D}$. If nodes include textual elements, such as names or descriptions, then features can be extracted using word embeddings, like Word2Vec [Mikolov et al., 2013] and GloVe [Pennington et al., 2014], or recent contextual language models like BERT [Devlin et al., 2019]. Edges and relations for every pair of connected nodes are defined as triples of the form (r_i^*, ρ, r_j^*), where ρ is a relation type, while r_i^* and r_j^* are nodes of the graph. On the other hand, a query graph Q^* consists of nodes $q^* \in Q^*$ that have to be matched to nodes in R^*.

Our goal is to correctly match a node $q^* \in Q^*$ to a node $r^* \in R^*$ such that $e_q = e_r$. In order to achieve this goal, we propose to process the graph with GNNs, as they allow leveraging relations between connected nodes, by exploiting the self-supervision provided by the structure of the reference graph. This process can be implemented in several ways. The first approach is to adopt the conventional classification setup. In that case, the GNN model can be extended with a final softmax layer, where the number of output nodes is equal to the number of classes (the number of unique entities in the set of entities E). However, this kind of models usually requires a large number of training examples for each class label. In contrast, for record linkage, the number of unique entities (classes) is typically much larger than in classification problems, and the number of records belonging to the same entity is usually small. Also, this approach requires re-training the model when new entities are added to the database.

The methodology we propose is instead based on learning distributed representations for every node in R^*, so that nodes representing the same entity are closer in the embedding space and far from irrelevant nodes. This can be achieved by applying *deep metric learning* methods, such as Siamese networks [Chopra et al., 2005] or triplet networks [Schroff et al., 2015]. R^* is used as training data for a GNN that produces an embedding vector of size

M for each node $r^* \in R^*$. We denote as γ_r the embedding of the node $r^* \in R^*$, whereas Γ_R represents the set of all the embeddings for every node in R^*. At inference time, for a node $q^* \in Q^*$ we compute an embedding vector $\gamma_q \in \mathbb{R}^M$ by applying the already trained GNN model. At this point, we have three potential scenarios:

1. q^* cannot be matched successfully to any node in R^*;
2. q^* can be matched to a single node $r^* \in R^*$;
3. q^* can be matched to multiple nodes that refer to the same entity.

For the first two cases, the rule to link q^* to a node in the reference graph is given by:

$$r_c^* = \arg\min_{r^* \in R^*} dist(\gamma_q, \gamma_r), \tag{7.1}$$

$$\mathcal{N}(q^*, R^*) = \begin{cases} r_c^* & \text{if } dist(\gamma_q, \gamma_{r_c^*}) < t \\ \perp & \text{otherwise,} \end{cases} \tag{7.2}$$

where $dist(\gamma_q, \gamma_r)$ is the distance (e.g., Euclidean) between vectors γ_q and γ_r, $r_c^* \in R^*$ is the closest node to q^* in the embedding space Γ_R, $t \in \mathbb{R}$ is a threshold on the distance, and $\perp$ means no matches were provided according to the given threshold t.

If multiple nodes can refer to the same entity, the predicted probability of a match to an entity e is proportional to the sum of the similarities between the embeddings of the k nearest neighbors $\Gamma_q^k \subseteq \Gamma_R$ and the query embedding γ_q:

$$P_k(e_q = e \mid q^*, R^*) \propto \sum_{\gamma_r \in \Gamma_q^k} \delta(e, e_r) \cdot (1 - dist(\gamma_q, \gamma_r)), \tag{7.3}$$

where $\delta(e, e_r)$ is 1 if $e = e_r$ and 0 otherwise. This is similar to K-nearest neighbors classification, where a new sample is labeled according to the majority vote within labels of its nearest neighbors in the training set. The workflow described in this section is depicted in Figure 7.2.

7.3.2 Siamese GCN for Record Linkage

In this section, we introduce a model capable of performing entity matching on graphs by combining the advantages of graph convolutional networks [Kipf and Welling, 2017] and Siamese networks [Bromley et al., 1993]. The model, termed Siamese Graph Convolutional Network (S-GCN), has the primary objective of learning discriminative hidden representations of nodes in a graph R^*, in such a way that they can then be used for further entity matching with previously unobserved data.

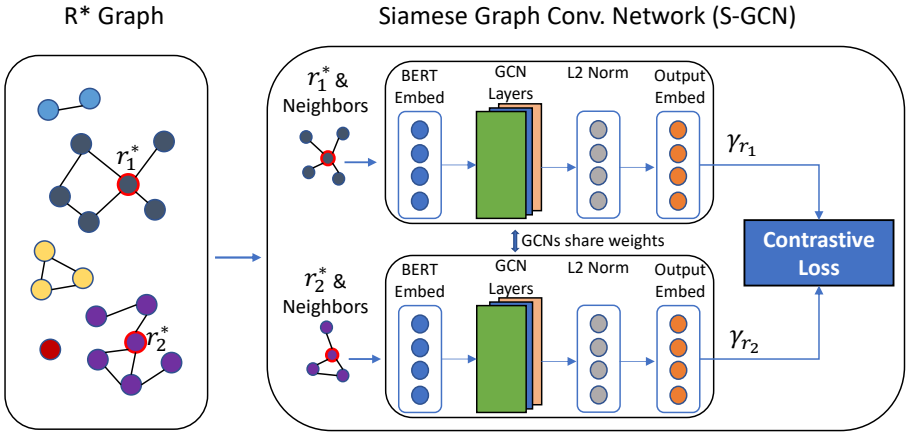


Figure 7.3: Architecture of the siamese graph convolution network.

GCNs [Kipf and Welling, 2017] are a type of GNN that share filter parameters among all of the nodes, regardless of their location in the graph. Every GCN layer can be regarded as a non-linear function:

$$Z^{(l+1)} = \phi(Z^{(l)}, A), \tag{7.4}$$

where l is the layer number and A is the adjacency matrix of the graph R^*. $Z^{(0)}$ is given by the initial feature matrix X_R. In our experiments, we employed a pre-trained BERT model [Devlin et al., 2019] as the input layer to create input features for nodes. The last GCN layer L produces the output embedding matrix for all nodes, which forms the set of embeddings Γ_R, where each embedding has dimension z. The propagation rule $\phi(\cdot)$ introduced in [Kipf and Welling, 2017] is defined as:

$$\phi(Z^{(l)}, A) = \sigma(\hat{D}^{-0.5}\hat{A}\hat{D}^{-0.5}Z^{(l)}W^{(l)}), \tag{7.5}$$

where $\hat{A} = A + I$, I is the identity matrix, $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$, and $W^{(l)}$ is a weight tensor of the l-th GCN layer.

Siamese neural networks are a popular approach to implementing one (or few) shot learning algorithms, so that items from the same category are close in the embedding space and far from items of different categories. A Siamese neural network is composed of two identical networks that share the same weights and accept two items as input. The first network outputs an encoding of the first item, and similarly the second one outputs an encoding of the second item. Our Siamese network is based on two identical GCNs. During the training period, the first GCN focuses on a given node r_1^* and its local neighborhood nodes $R_{r_1}^* \subset R^*$. This means that the GCN takes a subgraph around r_1^* as input and produces a vector of size z that finally forms the embedding vector γ_{r_1} of node r_1^*. The same procedure is repeated with the second GCN for node r_2^*, as shown in

Figure 7.3. Siamese networks are optimized with respect to the loss between their two outputs, namely γ_{r_1} and γ_{r_2}. The loss will be small for two nodes representing the same class (e. g., nodes of the same entity) and will be greater for nodes belonging to different classes. An instance of such a loss is the contrastive loss [Hadsell et al., 2006]:

$$\begin{aligned} &ContrastiveLoss(\gamma_{r_1}, \gamma_{r_2}) = \\ =&0.5 \cdot (1+y) \cdot dist(\gamma_{r_1}, \gamma_{r_2})^2 + \\ +&0.5 \cdot (1-y) \cdot \{max(0, m - dist(\gamma_{r_1}, \gamma_{r_2}))\}^2, \end{aligned} \tag{7.6}$$

where $y = 1$ if nodes (r_1^*, r_2^*) are from same class, and $y = -1$ otherwise, while m is a margin that is a non-negative number and indicates that dissimilar node pairs whose distance is beyond the margin will not penalize the network.

When the model is trained, one of the two Siamese networks (identical GCNs) is used for computing the output embedding matrix and forms Γ_R according to (7.4). It then sends the whole graph R^* as input to the GCN. Similarly, using the trained GCN, we can compute Γ_Q for an unobserved graph Q^* and perform entity matching in accordance to (7.2) and (7.3).

7.3.3 Siamese GCN with Subspace Learning

Training S-GCN with a large number of entities can suffer from poor local minima, as selecting informative dissimilar pairs of entities becomes extremely hard and computationally expensive. Inspired by [Wang et al., 2017], we adopt the subspace learning (SL) approach to train our Siamese GCN more effectively. In the following, we will refer to this network as S-GCN$_{SL}$.

A generally good way to train neural networks for distance metric learning is to select hard instances of dissimilar entities. In our case, the most informative pairs of dissimilar entities are given by nodes whose representations are close in the hidden space, but they refer to different entities. Therefore, sampling such difficult and informative entities leads the network to learn how to handle ambiguity and generalize better. However, as the number of entities grows, it becomes computationally intractable to find the most informative dissimilar pairs at every learning iteration. The idea of training S-GCN with SL is to initially generate the representation of all entities (e. g., via S-GCN) and then cluster all entity representations to form subspaces of the most similar entities. Then, the model is trained again over all subspaces, and dissimilar pairs are selected randomly within a subspace only. After each epoch, the updated model is used to generate new representations of the entities and the clustering and training steps are repeated again.

7.4 Datasets and Task Setup

To investigate the feasibility of our proposed approach, we set up a data integration task on a proprietary relational database that contains fine-grained information about business entities. This section describes how these data have been modeled as a graph and how the record-linkage task has been prepared and set up.

7.4.1 Graph Extraction

Each record in the database corresponds to a different company business location and is assigned a unique local identifier that can be used to track business entities precisely. Company entities represented in the database include different branches, subsidiaries and headquarters, each of which reports directly or indirectly to a given global identifier. Hence, companies related to the same global identifier can be grouped together and represented hierarchically as a single family. In addition, for each company entry, we have a textual name, and, optionally, up to three alternate names (aliases).

As mentioned in Section 7.2, we are interested in linking a company name to a family in the database. To this end, we represent families in the database as a graph, where each node is a company name. Relations between nodes are derived from the hierarchical representation of company families and from the aliases provided in the database. More precisely, a relation between two nodes exists if they are adjacent business entities in the hierarchy defined by the database, or if they are aliases of each other. Hence, the resulting graph consists of several disjoint connected components, each corresponding to a different family. The company database also provides other potentially valuable information, such as the precise location of each business entity. In this task, however, we do not model this additional information because we want to link companies based only on their names. This becomes useful in several real-world scenarios, as the company name that needs to be matched to the database may come from different structured or unstructured data sources, where this kind of additional information is usually not be available.

7.4.2 Data Preprocessing and Partitioning

Overall, the company database contains over 100M business entities located in approximately 200 countries. We focus our experiments on the records corresponding to companies located either in Switzerland or in the United States. This accounts for around 700k and 27M entities, respectively. We tested our approach on different graphs extracted from both datasets.

As discussed in Section 7.4.1, the graphs extracted from the company database contains a disjoint subgraph for each family. To investigate the feasibility of our approach, we

randomly remove nodes from the graph, and we then use the trained S-GCN model to predict the subgraph a previously unseen company name belongs to. More specifically, we partition our original dataset into training, validation, and test sets. The training set contains company names and specifies aliases and relations between entities in the same family. The validation and test sets include only the name of a company that is not represented in the training set and a class label that refers to the correct family. This means that the training set contains a graph for each company family, whereas the test set contains only *isolated nodes*. Modeling this kind of scenario with GNNs is challenging because nodes in the test set lack contextual information that would usually be derived and aggregated from neighboring nodes. Section 7.4.3 describes a possible way to mitigate this problem in our use case.

To provide enough training examples for each class label (i. e., for each company family), before partitioning the data into training, validation, and test sets, we first remove company families with less than $t \in \mathbb{N}$ companies. The threshold t is set to 5 for the graph that represents companies located in Switzerland, whereas it is set to 10 for the United States. This preprocessing step allows increasing the variability of the company names processed by the model at training time and guarantees that each family is adequately represented in the training, validation, and test sets.

As the record-linkage task we have set up works at the family level, we do not have an exact match at the record level from an entity in the test set to an equivalent entity in the training set. To address both issues, we partition the dataset into training, validation and test sets in such a way that names in the training set have at least some overlap with names in the test and validation sets.

More precisely, for companies located in Switzerland, we produce two different variants of the datasets, which we denote as overlap-1 and overlap-all. In the overlap-1 variant, we impose the constraint that each name in the validation and test sets has at least one representative word in common with at least one training example that belongs to the same family. This version of the datasets addresses the first problem of not having an exact match for company entities at the record level, while also mitigating the issue of non-overlapping names within the same family in the training set. Intuitively, this variant is meant to ensure that we can always find a record-level match between an entity in the test set and an entity from the same family in the training set.

On the other hand, in the overlap-all version, before partitioning the dataset into training, validation, and test sets, we first tokenize each company name into a list of words. We then identify a representative token for each family as the most frequent token among the names in that family. Next, we filter out all names that do not contain the representative word for their family, and we partition the remaining names into

	overlap-1	**overlap-all**	**US data**
# of families	900	700	28.6k
# of nodes	9k	4k	1.7M
# of edges	11k	4k	2.6M
avg. node degree	2.5	2	3.2
med. node degree	1	1	2
avg. # of samples per company family	10	6	116
med. # of samples per company family	5	4	32
# of val. nodes	2k	1.1k	145k
# of test nodes	2k	1.3k	153k
exact matches in test set	0%	0%	38%

Table 7.2: Statistics about the datasets.

non-overlapping training, validation, and test sets. This process addresses the issue of non-overlapping names within the same family, and produces cleaner datasets that are more suitable for training our S-GCN model. The overlap-1 and overlap-all variants remain still challenging for both traditional approaches and the proposed methodology, as will be demonstrated in Section 7.5. However, the datasets are specifically designed to provide enough commonalities to allow machine-learning approaches to generalize effectively to unseen records.

In order to generate the datasets that correspond to companies located in the United States (US data), we apply the same assumption used for producing the aforementioned overlap-1 variant. However, in this case, before partitioning the data into training, validation and test sets, we do not remove duplicate company names within the same family. This means that both the test and validation sets for the US data contain some names that appear in the training set. Overall, this design choice makes the datasets larger and more indicative of the performance of the model in a real-world scenario. Indeed, in practice, whether Q is a pre-existing relational database or a set of business entities extracted from news, the chances that a record $q \in Q$ matches exactly with another record $r \in R$ may not be negligible. In our use case, the procedure described above yields a test set where the percentage of exact matches against the training set is equal to 37.7% of the records. With a total of approximately 1.7 million nodes, the graph extracted from the US

data is particularly suitable to evaluate the scalability of our approach. In all the datasets introduced in this section, the training set includes 60% of the records in the database, while the remaining 40% is divided equally between the validation and test sets.

Table 7.2 presents selected statistics about the datasets introduced in this section.

7.4.3 Enriching the Graph with Short Names

As mentioned in Section 7.4.2, although the training set contains a connected component for each company family, the models are evaluated on a test set that contains company names as isolated nodes. To address this issue, we generate an additional alias for each company name in the test set, and link this alias to the real name. The enriched test set thus consists of small connected components of size two. This enhancement helps the GNN to better leverage the graph structure and the local neighborhood of nodes.

We generate additional names that are short or normalized versions of a company name. Short names are extracted using conditional random fields (CRFs) as described in [Gschwind et al., 2019]. CRFs are trained such that the most discriminative word or phrase in a company name is extracted as a short name. The importance of short names for linking company entities is also demonstrated in [Loster et al., 2017].

The same process can be applied to names in the training set in order to increase the size of the graph and provide more variability within the training data. Table 7.3 lists the number of distinct additional names that are added to the training, validation, and test sets by enriching the graphs with short names.

	overlap-1	**overlap-all**	**US data**
Training set	5.2k	1.9k	240k
Validation set	1.7k	0.9k	90k
Test set	1.8k	0.9k	90k

Table 7.3: Number of distinct additional names added to the graph.

7.5 Experimental Evaluation

This section provides an evaluation of the approach proposed in Section 7.3. We compare our approach against strong baselines, including a recent rule-based system [Gschwind et al., 2019] explicitly designed for the data described in Section 7.4. Then, we evaluate the quality of the embeddings learned by the S-GCN model.

7.5.1 Baselines

The approach outlined in Section 7.3 has been compared against two baselines. The first baseline is a record linkage system (RLS) for company entities, recently introduced in [Gschwind et al., 2019]. This system relies on a rule-based approach to score entities with different attributes, including the name of a company, the precise location, Standard Industry Codes (SIC)[1], and even short names, extracted as previously mentioned in Section 7.4.3.

We re-implemented the approach using the scoring function described in [Gschwind et al., 2019], which is based on a combination of weighted Levenshtein and Jaccard similarities between company names. RLS provides matches at the record level. Given a record $q \in Q$, the system outputs a record $r_q \in R$ such that

$$r_q = \arg\max_{r \in R}(score(r, q)). \tag{7.7}$$

where *score* is the aforementioned scoring function detailed in [Gschwind et al., 2019]. To adapt this method to our use case and perform matches at the family level, we simply define $RLS(q, R) = f_{r_q}$, where f_{r_q} is the family of the company represented by the record r_q, defined as in (7.7).

The second baseline is a multilayer perceptron (MLP) with a softmax classification layer on top, optimized with a cross-entropy loss function. The MLP takes BERT features as input for company names and produces a company family as output. Hyperparameters have been optimized manually on the validation set. To evaluate MLP on the datasets, we used the following setup:

- the number of fully connected layers varies from 1 to 3, depending on the dataset used;
- the learning rate is set to 10^{-3};
- every layer contains 500 hidden neurons;
- the dropout rate is 0.2;
- we apply a weight decay of $5 \cdot 10^{-5}$;
- the Adam optimizer is used for updating the weights of the model.

We train the network for up to 1000 epochs and stop the training process if the validation loss does not improve in the course of 50 epochs.

[1] `https://www.osha.gov/pls/imis/sicsearch.html`

7.5.2 Record Linkage Evaluation

We use the accuracy score, i. e. the proportion of correctly matched records in Q, to assess the performance of the proposed algorithms. As GCN is a well-established choice for classifying graph nodes, we designed a GCN with a final classification softmax layer as described in Section 7.3. Note that, since GCN-CLF is trained directly to perform classification, it must be retrained whenever a new company family enters the reference database R, whereas S-GCN is tolerant to new company families. S-GCN embeddings of nodes that represent a new company family will tend to group close to each other in the embedding space. This could allow the model to be extended to previously unseen families without having to train the network again from scratch.

Node features. Since nodes in our graphs correspond to company names, we can leverage well-established distributed representations of text in the form of word embeddings and contextual language models as meaningful initial features. As company names may be written in different languages, consist of abbreviations and even non-existing words, we used character-level n-grams and the multilingual BERT model. First we applied WordPiece tokenization [Devlin et al., 2019] to company names and then fed the resulting set of tokens through the trained BERT model with the reduced mean pooling strategy to obtain features for company names.

Record Linkage with S-GCN. S-GCN is trained on the reference graph R^* in a self-supervised way, so that the structure of the graph guides the training process. After the models have converged and all the reference nodes have been mapped into the S-GCN embedding space Γ_R, we compute the embeddings Γ_Q for all the nodes in Q^* by feeding them through the trained S-GCN. Then a one-nearest-neighbor (1-NN) search is applied to find the most similar vector $\gamma_r \in \Gamma_R$ to a node $q^* \in Q^*$:

$$r^* = SGCN(q^*, R^*) = \arg\min_{r_i^* \in R^*} dist(\gamma_q, \gamma_{r_i}). \tag{7.8}$$

As every vector γ_r refers to an entity e_r that in turn belongs to a family f_r, we can define the set of matches of records in Q against records in R as follows:

$$\mathcal{M}_{SGCN} = \{(q, r) \in Q \times R \mid SGCN(q^*, R^*) = r^*\}. \tag{7.9}$$

Then, given a set of matches $\mathcal{M}_\mathcal{N}$ produced by a model $\mathcal{N}$, we define the accuracy as:

$$Accuracy(\mathcal{M}_\mathcal{N}) = \frac{|\{(q, r) \in \mathcal{M}_\mathcal{N} \mid f_q = f_r\}|}{|\mathcal{M}_\mathcal{N}|}. \tag{7.10}$$

Training. For every epoch, we generate training pairs of similar and dissimilar nodes. Two nodes are considered to be similar only if they are connected by an edge. Thus,

some nodes cannot be joined into a pair of similar nodes, even if they belong to the same company family. This helps the network to cope with families where some nodes may have completely non-overlapping names. As an example, in the overlap-1 dataset, the nodes *Hotel Bellevue Palace* and *Infracore SA* belong to the same family. For every similar pair, we create dissimilar pairs by randomly sampling nodes from other company families.

Hyperparameters have been optimized on the validation sets using grid search. For GCN-CLF on overlap-1 and overlap-all, we used the following setup: the number of GCN layers is 3 and 1 respectively; the learning rate is 10^{-3}; every layer comprises 600 hidden neurons; the dropout probability is 0.2 and the weight decay is set to $5 \cdot 10^{-7}$ and $5 \cdot 10^{-5}$ respectively. To evaluate S-GCN, we used the following setup: the number of GCN layers is set 1; the learning rate is 10^{-3}; there are 600 hidden neurons in every layer; the dimensionality of the output embeddings is set to 300; the dropout value is 0.2; the weight decay is $5 \cdot 10^{-6}$; the margin value in the contrastive loss definition (7.6) is 0.45 and 0.35 respectively for overlap-1 and overlap-all; for every pair of similar nodes, two pairs of dissimilar nodes were sampled. We then use the Adam optimizer to update the weights of GCN-CLF and S-GCN. We train the networks for up to 1000 epochs and stop the training if the validation accuracy does not improve in the course of 50 epochs.

Results. We compared the GCN-CLF and S-GCN models against the baselines on the datasets introduced in Section 7.4.2. In addition, we studied the effect of enriching the graphs R^* and Q^* with short names, as described in Section 7.4.3. Overall, in all the datasets, the proposed graph-based approaches outperform the baselines. Analyzing the results on the overlap-1 dataset, we can easily see that enriching the graphs with short names allows improving accuracy by a large margin. Most notably, for S-GCN on overlap-1, accuracy rises by 13%. Adding short names increases the variability of training data and allows GNNs to effectively propagate information from short versions to the original names.

It is worth mentioning that record linkage on overlap-1 is an inherently difficult task for machine-learning approaches, because the dataset contains "noisy" families, as mentioned in Section 7.4.2. In this dataset, MLP performs better than GCN-CLF, but the best results are achieved by the S-GCN and S-GCN$_{SL}$ models, which reach an accuracy of 0.83 and 0.85 respectively. An explanation of the poor performance of GCN-CLF is given by the presence of connected non-overlapping company names in the KG. The network propagates this "noisy" information through edges thereby hampering the impact of "clean" samples.

S-GCN and S-GCN$_{SL}$ are less sensitive to noise within a company family and they outperform GCN-CLF by 13% and 15% on overlap-1 with short names. Recall that S-GCN is not trained directly for classification. Instead, it aims to put two similar nodes from

Algorithm	**overlap-1**	**overlap-all**
RLS	0.71	0.80
MLP	0.71	0.90
GCN-CLF	0.56	0.90
S-GCN	0.70	0.91
S-GCN$_{SL}$	0.75	0.92
RLS (+short names)	0.78	0.87
MLP (+short names)	0.75	0.91
GCN-CLF (+short names)	0.70	0.94
S-GCN (+short names)	0.83	0.93
S-GCN$_{SL}$ (+short names)	**0.85**	**0.95**

Table 7.4: Accuracy of data linkage algorithms on the overlap-1 and overlap-all datasets.

R^* closer within the embedding space Γ_R. We consider two nodes to be similar only if they are connected by an edge. Therefore, S-GCN tries to map to close embeddings only

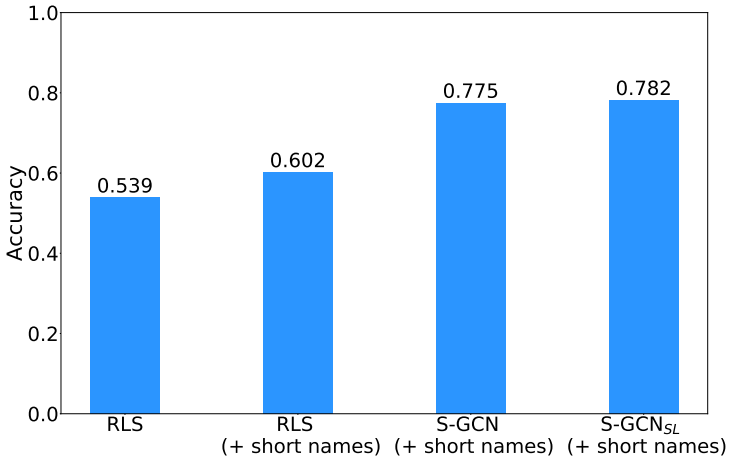


Figure 7.4: Accuracy on the US data.

connected nodes, without explicitly taking into account the other entities in the same family. Although some noise might come from neighboring nodes, S-GCN does not aim to learn a common pattern among all these entities at once, and, subsequently, it does not explicitly force grouping all entities in the same family.

The overlap-all dataset is "cleaner", as it only contains company names that share a given representative word with all the other entities in the same family. Hence all algorithms achieve a higher accuracy on this variant. The highest performance is reached by approaches based on the use of GNNs. Accuracy levels for S-GCN, S-GCN$_{SL}$ and GCN-CLF are comparable and consistently exceed the results obtained by the baselines.

Finally, we assessed the scalability of our approach by training our models a larger dataset that represents companies located in the United States. As mentioned in Section 7.4.2, the training set contains 1.7 million nodes, grouped in more than 28k families. On the one hand, the large size of the reference database is a potential threat for rule-based systems like RLS, but, on the other hand, dealing with a high number of classes poses hard challenges for approaches based on machine learning as well. We trained on this dataset our top-scoring systems, namely S-GCN and S-GCN$_{SL}$, both with short names. Figure 7.4 summarizes the results. As we can see, without short names RLS does not achieve a satisfactory accuracy and is able to identify the correct match only for 54% of the names in Q. The use of short names allows improving the performance of the system and achieving an accuracy of approximately 0.60. On this dataset, the S-GCN and S-GCN$_{SL}$ models achieve a comparable accuracy of 0.78, accounting for a significant improvement of 18% with respect to the best baseline, namely RLS with short names.

7.5.3 Accuracy-Coverage Tradeoff

Both the S-GCN and RLS approaches allow the parameters of the system to be tuned such that a company name in the test set is linked to a family in the training set only if the match could be identified with high confidence. In other words, for both systems we can specify a criterion to detect that a given company name in the test set could not be matched successfully to any family in the training set. This increases the probability that the results provided by the system are correct, at the expense of some company names possibly being left unclassified, even if a corresponding record exists in the reference database. Hence, we can identify a tradeoff between the accuracy and coverage of the system, where accuracy is defined by (7.10), and coverage is simply the number of either correct or wrong matches provided by the system over the total number of queries (i. e., the size of the test set):

$$Coverage(\mathcal{M}, Q) = \frac{|\mathcal{M}|}{|Q|}.$$

RLS [Gschwind et al., 2019] can be naturally tuned to avoid reporting wrong matches by simply setting a threshold $t \in [0, 1]$ on the value of the scoring function:

$$RLS_t(q, R) = \begin{cases} f_{r_q} & \text{if score}(r_q, q) \geq t \\ \perp & \text{otherwise,} \end{cases}$$

where *score* is the RLS scoring function and r_q is an RLS match for q against the database R, defined as in (7.7).

Similarly, we can tune our approach based on S-GCN to link a node $q^* \in Q^*$ to a family $f \in F$ only if all the $k \in \mathbb{N}$ nearest neighbors of the predicted embedding γ_q of q^* belong to family f. Formally,

$$SGCN_k(q^*, R^*) = \begin{cases} f & \text{if } e_r \in f, \forall \gamma_r \in \Gamma_q^k \\ \perp & \text{otherwise} \end{cases}$$

where $\Gamma_q^k \subseteq \Gamma_R$ is the set of the k nearest neighbors of the embedding predicted by our S-GCN model for q^*.

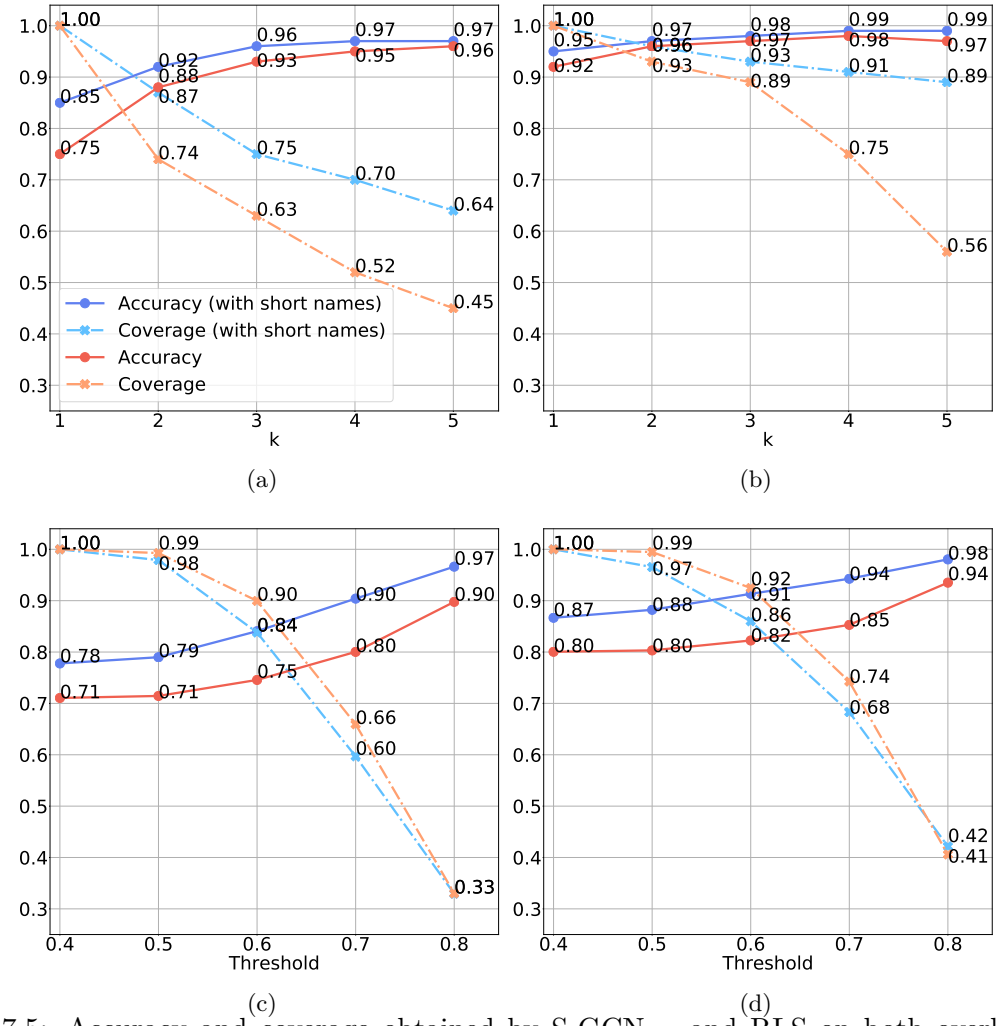


Figure 7.5: Accuracy and coverage obtained by S-GCN$_{SL}$ and RLS on both overlap-1 and overlap-all: a) S-GCN$_{SL}$ on overlap-1; b) S-GCN$_{SL}$ on overlap-all; c) RLS on overlap-1; d) RLS on overlap-all.

Figure 7.5 shows the accuracy and coverage achieved on overlap-1 and overlap-all by both RLS and S-GCN$_{SL}$ for different values of t and k, respectively. As outlined also in Table 7.4, on overlap-1, at a coverage value of 1.00, without using short names, the accuracy of S-GCN$_{SL}$ is only moderately higher than that of RLS. Clearly, increasing the values of k and t allows improving the accuracy of both systems at the expense of decreasing coverage. However, for S-GCN we can improve accuracy while keeping coverage relatively high compared to RLS. Setting $k = 3$ is sufficient to exceed an accuracy of 90% for S-GCN$_{SL}$, on overlap-1 without short names. To achieve a comparable accuracy of 0.90 for RLS, we have to increase the threshold to 0.8, resulting in a very low coverage of 0.33.

Using short names allows us to boost the performance of S-GCN$_{SL}$ even further. If we aim to cover all the records in the test set, the accuracy of S-GCN reaches 0.85, whereas that of RLS reaches 0.78. Moreover, setting $k = 2$ allows us to raise the accuracy to 0.92 with a high coverage of 0.87. On the other hand, RLS can only reach a comparable accuracy of 0.90 by dropping coverage to 0.60.

The performance of our model is much better on overlap-all, as this dataset is more suitable for learning. We observe a similar pattern for short names as their usage improves the accuracy for both systems. However, coverage for RLS drops faster for comparable improvements in accuracy. Most notably, S-GCN$_{SL}$ can achieve an almost perfect accuracy of 0.98 with a high coverage of 0.93. Thus, we can set the parameters of the system such that we are highly confident that predictions are correct, while obtaining a match in more than 90% of the cases. On the other hand, reaching a very high accuracy of 0.98 with RLS requires dropping coverage to 0.42.

7.5.4 Learning Company Embeddings

In this section, we study the consistency and the quality of learned S-GCN embeddings. We employ *silhouette scores* [Rousseeuw, 1987] as a measure of how well the entities of company families are clustered in the S-GCN embedding space. A silhouette value of an entity measures how close a company entity is to its family in the projection. The silhouette value is defined in a range of $[-1, 1]$, where a high score indicates that the entity is well coordinated with its company family. If the average silhouette score (over all entities) has a positive value, then families are disjoint on average, whereas if the score is negative, families are overlapping in the embedding space. We computed the silhouette score for S-GCN where the features are initialized with the embeddings from the BERT model and the pre-trained BERT model on the two datasets overlap-1 and overlap-all. The results are shown in Table 7.5. Average silhouette scores for S-GCN are always positive and are higher that the results of the sole BERT model demonstrating that S-GSN is able to produce more discriminating features by using additional information flow of the company graph.

To demonstrate the results visually, we randomly selected ten company families from the embeddings produced by our S-GCN model (on overlap-all with short names) and projected 300 embedding dimensions into 2D with t-SNE [van der Maaten and Hinton, 2008]. Figure 7.6 shows the 2D projection of the data, where the dots correspond to reference records from R (training examples), colors indicate company families, and stars represent corresponding records from Q (test records). We see that q records from the database Q (stars) can be matched to their families even in 2D space.

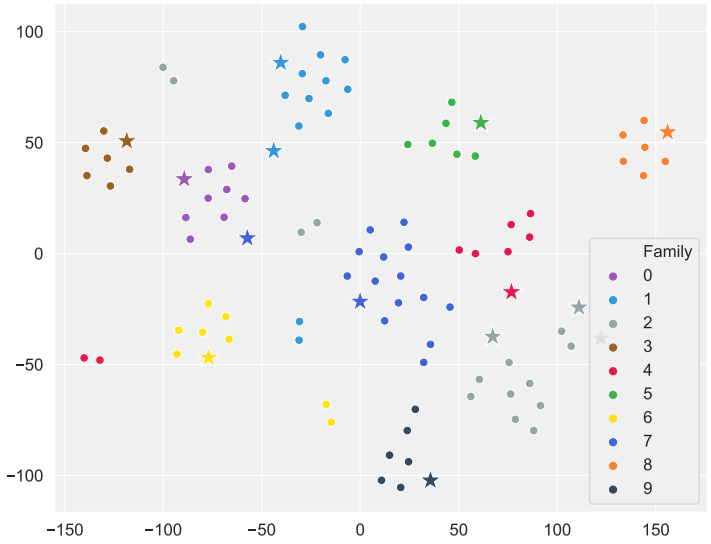


Figure 7.6: 2D representation of 10 randomly selected company families computed by S-GCN, where dots represent records $r \in R$, while stars refer to records $q \in Q$.

	Avg. silhouette score	
	BERT + S-GCN	**only BERT**
overlap-1	0.22	−0.12
overlap-all	0.63	0.01

Table 7.5: Average silhouette score of company families.

7.5.5 Conclusion

Although the experimental results are promising, we envision several challenges that need to be addressed for a proper application of GNNs to data integration tasks. This section lists some of the main problems that have to be investigated and require further research.

Graph extraction. Generating a graph-structured representation of the data is a crucial and time-consuming process, with a potentially strong impact on subsequent design choices as well as on the final performance of the model. Relational databases usually provide many different connections between entities, and can typically be modeled as a graph in several different ways. Defining exactly what kind of entities should be represented as a node and what kind of relations should be included as edges strongly affects the way information is propagated between the neighboring nodes during training.

Moreover, as noted in [Xirogiannopoulos and Deshpande, 2017], extracting a graph from

a relational database has several other potential applications, thanks to the insights that can be drawn from explicit modeling of entities and their connections. Some automatic or semi-automatic approaches have been proposed in the literature. For instance, a domain-specific language (DSL) based on Datalog is used in [Xirogiannopoulos and Deshpande, 2017] to specify graph-extraction queries. Recently, an approach to cleaning raw tabular data automatically and generating linked data has been introduced in [Sukhobok et al., 2016]. Additionally, [Arenas et al., 2012] describes a direct mapping from the data in a relational database to a knowledge graph. The algorithms defined in [Arenas et al., 2012] can even be applied to multiple tables, where foreign keys are used to establish a reference from a given row in a table to exactly one row in a potentially different table.

In spite of these results, the process of extracting graph representations of relational databases is still challenging and labor intensive, and is often performed manually. The need for automatic approaches is an even stronger requirement for unstructured data. Some recent work in this area includes [Gangemi et al., 2016] and [Das et al., 2019].

Handling different relations. As previously discussed in Section 7.4.1, in our application we only use a single relation type to connect both aliases of the same company and companies belonging to the same family. More precisely, in our use case, an edge in the graph R^* is a pair of the type (r_i^*, r_j^*), where r_i^* and r_j^* are adjacent nodes in the graph. However, in several realistic large-scale knowledge bases, this assumption is usually restrictive. Relations between two entities in a KG can be of type $\tau \in \mathcal{T}$, where $\mathcal{T}$ is the set of all possible relations between two entities in the graph. In this case, edges are not represented as pairs, but are modeled as triples of the form (r_i, τ, r_j). This poses some challenges related to processing large-scale relational data with GNNs. Recently, a variation of GCN that is capable of making relation-specific transformations based on the type and direction of an edge was introduced in [Schlichtkrull et al., 2018]. The model, termed Relational GCN (R-GCN), is a special case of the message-passing neural network proposed in [Gilmer et al., 2017], as it propagates messages while taking into account relation types by learning a different weight matrix for each $\tau \in \mathcal{T}$. In practice, this poses several challenges when dealing with highly multi-relational data. Indeed, the number of parameters in the model grows with the number of relations in the graph. This can result in very large models, or even lead to overfitting on rare relations. Some regularization strategies to address this issue have been introduced in [Schlichtkrull et al., 2018]. These techniques allow the total number of parameters to be limited, either by sharing parameters among weight matrices or enforcing sparsity.

Crowdsourced data augmentation. Some nodes in R^* or Q^* might have a small degree of incoming edges. Artificially generating aliases for a node with a low degree and connecting them to the original node might lead to a better generalization ability

of GNNs and consequently improve its accuracy. Information from aliases propagates towards the original node, where it accumulates with the node information resulting in the output embedding. In our use case, we improved the results greatly by generating short company names from their original names (see Section 7.4.3). To this end, we can leverage crowdsourcing micro-tasking for creating aliases via the human perception of real entities. This will enrich training data by increasing the variability of entities, therefore allowing to generate more informative embeddings for target nodes.

Chapter 8

Conclusions and Future work

8.1 Contributions

In this closing chapter, we will outline our contributions, provide answers to the thesis research questions, discuss limitations as well as future work.

Our contribution can be summarized as the following:

1. We confirmed initial findings that crowdsourcing is feasible for paper screening in Systematic Literature Reviews. Furthermore, crowd-based screening can be done with high precision and cost figures that are much lower with respect to what authors spend today in terms of effort [Krivosheev et al., 2017, 2018b].

2. The model that provides users with a list of options and error-cost estimates for each option in crowdsourcing classification. The main point of this probabilistic model is to do not waste money blindly, rather get an insight into quality outcomes for specific values of task parameters (e. g., budget, number of votes per item, number of test questions, loss for false positive and false negative errors). Hence, a user can choose the most suitable strategy for him/her and the data at hand [Krivosheev et al., 2017].

3. The probabilistic crowdsourcing model for multi-predicate item classification that iteratively learns statistical parameters of the problem to i) identify easy item-predicate to classify first, ii) limit spendings without compromises on quality, iii) stop crowdsourcing an item if the workers cannot provide reliable votes on it (e. g., due to high difficulty of the item) [Krivosheev et al., 2018b].

4. A formulation of the general problem of hybrid crowd-machine classificaiton. Methods that combine machine learning and human classifiers to achieve higher classification accuracy for a unit of cost, where we are given a set of machine classifiers of unknown

accuracy for the problem at hand [Krivosheev et al., 2018a]. We proposed and discussed a set of budget allocation heuristics that can be adapted for efficiently balancing between learning and exploitation (of machine classifiers) in the hybrid crowd-machine classification task (Chapter 5).

5. The algorithm for detecting and preventing confusions of crowd observations on similar classes of items. By using the proposed algorithm task designers can detect confusions early in crowdsourcing process and modify the crowdsourcing task to prevent confusions in the next iterations of the task execution (Chapter 6).

6. The Siamese Graph Convolution Network architecture for modeling and matching entities from structured data (e. g., relational databases) as well as unstructured data sources (e. g., news articles). The proposed entity matching network is deployed as a RESTful web service that accepts a company name as input and provides back the corresponding business family in [Dun & Bradstreet Inc] database (Chapter 7). Demo video is available online[1].

7. Due to the lack of datasets (with actual crowd votes) for multi-predicate classification and detection confused observations, we contributed six open sourced datasets for multi-predicate crowd/crowd-machine classification, and confusion detection algorithms[2] [3].

More specifically, in response to our thesis research questions.

To create adaptive crowd strategies for optimization quality vs. cost trade-off in classification with a different loss of errors **(RQ1).**

We proposed and evaluated a set of strategies for crowd-based binary classification, and multi-predicate classification where items need to satisfy a conjunction of predicates. We analyzed *single-run* strategies including majority voting, expectation-maximization based algorithms (such as Dawid&Skin, Truth Finder, etc.). The obvious improvement to the aforementioned algorithms is to consider the loss ratio, i. e., how much a false negative is more "harmful" than a false positive. This time, we minimize the loss by selecting optimal parameters for the classification function.

We proposed a list of cost-effective *multi-run* strategies. The most notable of them is the adaptive shortest-run algorithm for multi-predicate classification that defines an *individual* strategy for each item to be labeled, aimed at identifying the shortest path to decision. With each new coming vote, we update our estimation of the parameters and identify for each item the predicate for which a vote takes us closer or past our exclusion

[1] https://youtu.be/LDzCZNrCm3o

[2] https://github.com/Evgeneus/screening-classification-datasets

[3] https://github.com/Evgeneus/fuzzy-data-fusion/tree/master/data

threshold. We also can rank papers based on this, and identify the low hanging fruits. We validated both applicability and results of the aforementioned approaches on a number of synthetic and real-word crowdsourced datasets, and discussed properties of the problem and algorithms that we believe to be generally of interest for classification problems. Our experiments demonstrated that the multi-run algorithms (especially shortest-run) significantly outperform a number of non-adaptive approaches in terms of cost and accuracy.

Different algorithms can be used to spot the parameters of the crowdsourcing task and the best algorithm we identified is based on a multi-run strategy. We proposed a probabilistic model for reasoning over the problem, for tuning the parameters of crowdsourcing tasks to minimize errors, and for providing users with an estimated curve of errors vs. budget trade-off for different task parameters. Note the users are not required to fill in parameters, they only need to state their subjective preferences on how much budget they have and on how "painful" they consider a false negative to be, for their specific problem. All the other parameters can be entered if available in the form of beliefs, but in the absence of a belief, the algorithm proceeds to estimate them.

Last but not least, we performed more than 30 crowdsourcing experiments, which provided many insights on task design and nature of the problems, such as how the problem should be framed to increase participation and reduce errors, how to make the task more approachable, as well as actual pay scales considered acceptable by the community.

To build efficient hybrid crowd-machine classifiers under a limited budget **(RQ2).**

We proposed several methods for multi-predicate classification that combines machine learning and human classifiers to achieve a high level of classification accuracy for a fraction of budget. We believe the main benefit lies in the ability of the algorithms to be robust to both the characteristics of the problem (such as predicate that are hard for the crowd to classify) and to weak machine classifiers. In both cases, the algorithms work out to leverage predicates, items, and machine classifiers (over specific predicate and items) and build models of classifiers to make the most of what can be classified with the lowest number of crowd votes as possible, leaving the most difficult items for other classification methods (such as expert classification) without compromising on quality.

We validated our approaches on synthetic and crowdsourced datasets in different domains. The results showed that even a small fraction of budget invested in machine training leads to improvement in cost and accuracy with respect to the state-of-the-art screening algorithm available. Increasing the budget allocation to learning up to 50% increases accuracy but also cost, and from that point on the improvements vary. We also observed that a difference with respect to crowd only classification become more manifest as crowd accuracy grows over 0.7-0.8 as this leads to "good" training data. Adding a sprinkle of machine learning to crowd classification and a sprinkle of crowd to machine

classification leads to high accuracy and better results in finite pool problems, provided that the budget vs. pool size ratio is such that we can collect crowd votes rather than use machines only.

To improve the quality of observations in crowd-based classification **(RQ3).**

From our crowdsourcing experiments, we recognized that subtle form of crowd errors is due to confusion of observations, that is, crowd workers (including diligent ones) that confuse items of a class i with items of a class j, either because they are similar or because the task description has failed to explain the differences. We show that confusion of observations can be a frequent occurrence in many tasks and that such confusions induce significant noise in datasets. We proposed MCMC-C algorithm for detecting such confusions, leveraging an inference procedure based on Markov Chain Monte Carlo sampling. We have shown that MCMC-C is an effective to detect confusion over a variety of datasets (up to 98% in accuracy), and that once confusion is detected, it can then be prevented to a large extent with simple modifications to the task. The algorithm scales well to large datasets (both in items and workers). We showed that it is also possible to automatically correct confused assignments of labels during the data fusion process what leads to improvement in crowd votes aggregation.

Efficiently link entities from structured and unstructured data sources **(RQ4).**

We introduced a general approach to modeling and integrating entities from structured and unstructured sources. Our approach is designed to explicitly model and leverage relations between entities, thereby using all available information and preserving as much context as possible. This is achieved by combining siamese and graph neural networks to propagate information between connected entities and support high scalability. We evaluated our method on the task of matching data about business entities, and we demonstrated that it outperforms standard state-of-the-art rule-based systems, as well as other deep learning approaches that do not use graph-based representations by up to 18% in accuracy.

8.2 Limitations

Our work on crowd and hybrid crowd-machine classification has several limitations. As the models (and real-life scenarios) have many parameters, a more in-depth discussion and analysis are needed. We have not discussed and analyzed the impact of clarity and of ongoing tests submitted to workers who pass the test phases. A detailed comparison with actual errors performed when experts decide inclusion and exclusion is also needed for a comprehensive evaluation of the approach in the domain of literature reviews. Algorithms still have rooms for improvement, for example in terms of finding the optimal number of

items to consider for the initial run of the multi-run strategies or to research an alternative way for parameter estimation (such as crowd accuracy and the selectivity of predicates). Apart from the points listed above, the general problem of the screening phase of SLRs and multi-predicate classification incorporates different areas that impact results, while our main focus lays within crowd and machine classification. Though we experimented with task design, a more in-depth study is needed for SLR classification, e. g., study the effect of highlighting key-words in abstracts [Ramírez et al., 2019]. Although the evaluation mainly has focused on the screening phase of literature reviews we see the approach as applicable to many multi-predicate screening problems, however, an extensive evaluation of the limits of the approaches and the kinds of problems to which it applies is needed. Therefore, an implementation of the operating system for the screening is required. It will allow us to perceive the quality of crowd abstract screening in scale for a variety of reviews and domains (also possible to use already published reviews).

A limitation of the confusion detection approach is that cluster identification is either manual or done via exploration of all possible clusters. In terms of identifying clusters, one option is to do so manually, however, MCMC-C is efficient and can iterate over possible pairs of labels. We tested this for our crowdsourced datasets and identify confusions with a percentage very close to what reported for manually selected ones. While the algorithm works fine in the majority of real-life problems where the number of classes is in the single or double-digit range and we look for clusters of size 2, but it becomes computationally expensive in other cases. Therefore, we need strategies to identify possible clusters beforehand so that we can later apply MCMC-C.

Although the S-GCN results are promising in the business entity matching task, we envision several limitations that need to be addressed for a proper application of GNNs to entity linkage. In our work, we did not cover in-depth the effect of generating a graph-structured representation of a database at hand, and how this influences to the information flow through edges and final results. Another limitation of the current S-GCN model is that we utilized only one type of relations in the graph, however, in many real-life problems we operate over graphs with different types of relations as well. Further, there is a need to test the applicability of the proposed approach on a variety of datasets from different domains and characteristics.

8.3 Future Work

We are planning to explore variations of adaptive policies in balancing learning vs. exploitation trade-off in hybrid classification. Particularly, we will attempt to compute reward in multi-armed bandit style and design an algorithm that will learn an optimal

learning vs. exploitation policy in reinforcement learning fashion fitted to a dataset and crowd workers at hand.

An interesting thread of analysis is to see if crowd workers can, besides labeling (item, predicate) pairs, actually identify features that machines can then leverage for the problem at hand.

Another promising problem is the extent to which learning can be transferred across finite pool problems of the same kind (for example, different literature reviews), especially in terms of identifying general approaches and methodologies that can be adopted in different domains.

Future work in entity matching direction will aim at exploring several aspects that are both methodological and architectural. On the methods side, we aim at providing guidance for cases where attributes in a record are better represented as node features rather than separate nodes. Also, we will explore different training strategies and GNN architectures. To improve the quality of linkage, we can generate extra aliases to graph nodes. To this end, we will utilize crowdsourcing to create additional human-generated aliases. This will enrich training data by increasing the variability of entities, therefore allowing to generate more informative embeddings for target entities.

Bibliography

Aggarwal, C.; Kong, X.; Gu, Q.; Han, J., and Yu, P. *Active Learning: A Survey, Data Classification: Algorithms and Applications*. CRC Press, 2014.

Alonso, O. and Marchionini, G. *The Practice of Crowdsourcing*. Morgan & Claypool, 2019. ISBN 9781681735245. URL `https://ieeexplore.ieee.org/document/8726280`.

Amershi, Saleema; Weld, Dan; Vorvoreanu, Mihaela; Fourney, Adam; Nushi, Besmira; Collisson, Penny; Suh, Jina; Iqbal, Shamsi; Bennett, Paul N.; Inkpen, Kori; Teevan, Jaime; Kikin-Gil, Ruth, and Horvitz, Eric. Guidelines for human-ai interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 3:1–3:13, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-5970-2. doi: 10.1145/3290605.3300233. URL `http://doi.acm.org/10.1145/3290605.3300233`.

Arenas, Marcelo; Bertails, Alexandre; Prud'hommeaux, Eric, and Sequeda, Juan. A Direct Mapping of Relational Data to RDF. `https://www.w3.org/TR/rdb-direct-mapping/`, September 2012.

Arkes, H. and Blumer, C. The psychology of sunk cost. *Organizational Behavior and Human Decision Processes*, 35(1), 1985.

Arora, Rajesh Kumar. *Optimization: algorithms and applications*. CRC Press, 2015.

Auer, Peter; Cesa-Bianchi, Nicolo; Freund, Yoav, and Schapire, Robert E. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

Avnur, Ron and Hellerstein, Joseph M. Eddies: Continuously adaptive query processing. *SIGMOD Rec.*, 29(2): 261–272, May 2000. ISSN 0163-5808. doi: 10.1145/335191.335420. URL `http://doi.acm.org/10.1145/335191.335420`.

Aydin, Bahadir Ismail; Yilmaz, Yavuz Selim; Li, Yaliang; Li, Qi; Gao, Jing, and Demirbas, Murat. Crowdsourcing for multiple-choice question answering. In *AAAI*, pages 2946–2953, 2014.

Azmy, Michael; Shi, Peng; Lin, Jimmy, and Ilyas, Ihab F. Matching entities across different knowledge graphs with graph embeddings. *CoRR*, abs/1903.06607, 2019.

Babu, Shivnath; Motwani, Rajeev; Munagala, Kamesh; Nishizawa, Itaru, and Widom, Jennifer. Adaptive ordering of pipelined stream filters. In *Proceedings of ACM SIGMOD*. ACM, 2004.

Baram, Yoram; El-Yaniv, Ran, and Luz, Kobi. Online choice of active learning algorithms. *J. Mach. Learn. Res.*, 5:255–291, December 2004. ISSN 1532-4435.

Battaglia, Peter W.; Hamrick, Jessica B.; Bapst, Victor; Sanchez-Gonzalez, Alvaro; Zambaldi, Vinícius Flores; Malinowski, Mateusz; Tacchetti, Andrea; Raposo, David; Santoro, Adam; Faulkner, Ryan; Gülçehre, Çaglar; Song, Francis; Ballard, Andrew J.; Gilmer, Justin; Dahl, George E.; Vaswani, Ashish; Allen, Kelsey; Nash, Charles; Langston, Victoria; Dyer, Chris; Heess, Nicolas; Wierstra, Daan; Kohli, Pushmeet; Botvinick, Matthew; Vinyals, Oriol; Li, Yujia, and Pascanu, Razvan. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL `http://arxiv.org/abs/1806.01261`.

Beygelzimer, Alina; Dasgupta, Sanjoy, and Langford, John. Importance weighted active learning. ICML '09, pages 49–56, 2009. ISBN 978-1-60558-516-1.

Beygelzimer, Alina; Langford, John; Li, Lihong; Reyzin, Lev, and Schapire, Robert. Contextual bandit algorithms with supervised learning guarantees. In Gordon, Geoffrey; Dunson, David, and Dudk, Miroslav, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15. PMLR, 2011.

Bragg, Jonathan; Mausam, , and Weld, Daniel S. Optimal testing for crowd workers. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 966–974, Richland, SC, 2016. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-4239-1. URL `http://dl.acm.org/citation.cfm?id=2937029.2937066`.

Brin, Sergey and Page, Lawrence. The anatomy of a large-scale hypertextual web search engine. In *WWW*, pages 107–117, 1998.

Bromley, Jane; Guyon, Isabelle; LeCun, Yann; Säckinger, Eduard, and Shah, Roopak. Signature verification using a "siamese" time delay neural network. NIPS'93, pages 737–744, 1993.

Brown, Andrew W. and Allison, David B. Using crowdsourcing to evaluate published scientific literature: Methods and example. *Plos One*, 9(7), 2014.

Bruss, Thomas. Sum the odds to one and stop. *Annals of Probability*, 28(3), 2000.

Callaghan, William; Goh, Joslin; Mohareb, Michael; Lim, Andrew, and Law, Edith. Mechanicalheart: A human-machine framework for the classification of phonocardiograms. *Proc. ACM Hum.-Comput. Interact.*, 2(CSCW): 28:1–28:17, November 2018. ISSN 2573-0142. doi: 10.1145/3274297.

Chang, Joseph Chee; Amershi, Saleema, and Kamar, Ece. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 2334–2346, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3026044. URL `http://doi.acm.org/10.1145/3025453.3026044`.

Cheng, Justin and Bernstein, Michael S. Flock: Hybrid crowd-machine learning classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, pages 600–611, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-2922-4. doi: 10.1145/2675133.2675214.

Cherman, Everton Alvares; Tsoumakas, Grigorios, and Monard, Maria-Carolina. Active learning algorithms for multi-label data. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 267–279. Springer, 2016.

Chopra, Sumit; Hadsell, Raia, and LeCun, Yann. Learning a similarity metric discriminatively, with application to face verification. CVPR '05, pages 539–546, 2005.

Chu, Xu; Ilyas, Ihab F.; Krishnan, Sanjay, and Wang, Jiannan. Data cleaning: Overview and emerging challenges. In *SIGMOD '16*, 2016.

Créquit, Perrine; Trinquart, Ludovic; Yavchitz, Amélie, and Ravaud, Philippe. Wasted research when systematic reviews fail to provide a complete and up-to-date evidence synthesis: the example of lung cancer. *BMC Medicine*, 14(1):8, 2016. ISSN 1741-7015. doi: 10.1186/s12916-016-0555-0. URL `http://dx.doi.org/10.1186/s12916-016-0555-0`.

Cybenko, George. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Dai, Peng; Lin, Christopher H.; Mausam, , and Weld, Daniel S. Pomdp-based control of workflows for crowdsourcing. *Artif. Intell.*, 202(1):52–85, September 2013. ISSN 0004-3702. doi: 10.1016/j.artint.2013.06.002.

Das, Rajarshi; Munkhdalai, Tsendsuren; Yuan, Xingdi; Trischler, Adam, and McCallum, Andrew. Building dynamic knowledge graphs from text using machine reading comprehension. *arXiv preprint arXiv:1810.05682*, 2018.

Das, Rajarshi; Munkhdalai, Tsendsuren; Yuan, Xingdi; Trischler, Adam, and McCallum, Andrew. Building dynamic knowledge graphs from text using machine reading comprehension. In *ICLR*, 2019.

Dawid, A. P. and Skene, A. M. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C Applied Statistics*, 28(1), 1979.

Dempster, A. P.; Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1), 1977.

Deshpande, Amol; Ives, Zachary, and Raman, Vijayshankar. Adaptive query processing. *Foundations and Trendsin Databases*, 1(1):1–140, 2007. ISSN 1931-7883. doi: 10.1561/1900000001. URL `http://dx.doi.org/10.1561/1900000001`.

Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton, and Toutanova, Kristina. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

Dietterich, Thomas G. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2), 2000.

Dong, Xin Luna and Rekatsinas, Theodoros. Data integration and machine learning: A natural synergy. SIGMOD '18, pages 1645–1650, 2018.

Dong, Xin Luna; Berti-Equille, Laure, and Srivastava, Divesh. Integrating conflicting data: the role of source dependence. *VLDB*, 2:550–561, 2009.

Dong, Xin Luna; Berti-Equille, Laure, and Srivastava, Divesh. Data fusion : Resolving conflicts from multiple sources. In *Procs of WAIM2013*. Springer, 2013. ISBN 9783642362576. doi: 10.1007/978-3-642-36257-6.

Dumitrache, Anca; Inel, Oana; Timmermans, Benjamin; Martinez-Ortiz, Carlos; Sips, Robert-Jan; Aroyo, Lora, and Welty, Christopher A. Empirical methodology for crowdsourcing ground truth. *CoRR*, abs/1809.08888, 2018.

Dun & Bradstreet Inc. Dun & Bradstreet Company. `http://www.dnb.com`, 2019.

Džeroski, Saso and Ženko, Bernard. Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54(3):255–273, Mar 2004. ISSN 1573-0565.

Eickhoff, Carsten. Cognitive biases in crowdsourcing. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 162–170, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5581-0. doi: 10.1145/3159652.3159654. URL `http://doi.acm.org/10.1145/3159652.3159654`.

Eickhoff, Carsten and de Vries, Arjen P. Increasing cheat robustness of crowdsourcing tasks. *Information retrieval*, 16(2):121–137, 2013.

Eveleigh, Alexandra; Jennett, Charlene; Blandford, Ann; Brohan, Philip, and Cox, Anna L. Designing for dabblers and deterring drop-outs in citizen science. In *Procs of CHI 2014*. ACM Press, 2014.

Franklin, Michael J.; Kossmann, Donald; Kraska, Tim; Ramesh, Sukriti, and Xin, Reynold. Crowddb: Answering queries with crowdsourcing. In *Proceedings of ACM SIGMOD*. ACM, 2011.

Frey, Bruno S. and Jegen, Reto. Motivation crowding theory. *Journal of Economic Surveys*, 15(5), 2001.

Ganea, Octavian-Eugen and Hofmann, Thomas. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2619–2629, 2017.

Gangemi, Aldo; Presutti, Valentina; Reforgiato Recupero, Diego; Nuzzolese, Andrea; Draicchio, Francesco, and Mongiovi, Misael. Semantic web machine reading with fred. *Semantic Web*, 8:1–21, 09 2016. doi: 10.3233/SW-160240.

Ganti, Ravi and Gray, Alexander. Upal: Unbiased pool based active learning. In Lawrence, Neil D. and Girolami, Mark, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 422–431. PMLR, 2012.

Garneau, Nicole L; Nuessle, Tiffany M; Sloan, Meghan M; Santorico, Stephanie A; Coughlin, Bridget C, and Hayes, John E. Crowdsourcing taste research: genetic and phenotypic predictors of bitter taste perception as a model. *Frontiers in integrative neuroscience*, 8:33, 2014.

Getoor, Lise and Machanavajjhala, Ashwin. Entity resolution: Theory, practice & open challenges. *PVLDB*, 2012.

Giancola, Michael; Paffenroth, Randy, and Whitehill, Jacob. Permutation-invariant consensus over crowdsourced labels. In *HComp*, 2018.

Gilmer, Justin; Schoenholz, Samuel S.; Riley, Patrick F.; Vinyals, Oriol, and Dahl, George E. Neural message passing for quantum chemistry. ICML'17, 2017.

Gomes, Ryan; Welinder, Peter; Krause, Andreas, and Perona, Pietro. Crowdclustering. In *Procs of Nips 2011*, 2011.

Gori, M.; Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 2, pages 729–734 vol. 2, July 2005. doi: 10.1109/IJCNN.2005.1555942.

Gottapu, Ram Deepak; Dagli, Cihan, and Ali, Bharami. Entity resolution using convolutional neural network. *Procedia Computer Science*, 95:153 – 158, 2016. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2016.09.306. URL `http://www.sciencedirect.com/science/article/pii/S1877050916324796`. Complex Adaptive Systems Los Angeles, CA November 2-4, 2016.

Grant, M.J. and Booth, A. A typology of reviews: an analysis of 14 review types and associated methodologies. *Health Info Libr J*, 26(2):91–108, 2009.

Green, JPT Higginsand S. *Cochrane Handbook for Systematic Reviews of Interventions Version 5.1.0*. The Cochrane Collaboration, 2011. Available from www.handbook.cochrane.org.

Gschwind, Thomas; Miksovic, Christoph; Minder, Julian; Mirylenka, Katsiaryna, and Scotton, Paolo. Fast record linkage for company entities. *IEEE Big Data 2019, December 9-12, 2019, Los Angeles, CA, USA*, 2019.

H. Riesch H, C. Potter. Citizen science as seen by scientists: Methodological, epistemological and ethical dimensions. *TUGBoat*, 23(1), 2014.

Hadsell, Raia; Chopra, Sumit, and LeCun, Yann. Dimensionality reduction by learning an invariant mapping. CVPR '06, 2006.

Haidich, A. Meta-analysis in medical research. *Hippokratia*, 14(Suppl 1):29–37, 2010.

Hamilton, Will; Ying, Zhitao, and Leskovec, Jure. Inductive representation learning on large graphs. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 1024–1034. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.pdf`.

Harris, Joshua D; Quatman, Carmen E; Manring, MM; Siston, Robert A, and Flanigan, David C. How to write a systematic review. *The American journal of sports medicine*, 42(11):2761–2768, 2014.

Heinecke, Shelby and Reyzin, Lev. Crowdsourced pac learning under classification noise. *ArXiv*, abs/1902.04629, 2019.

Hellerstein, Joseph M. and Stonebraker, Michael. Predicate migration: Optimizing queries with expensive predicates. In *Proceedings of ACM SIGMOD*. ACM, 1993.

Henderson, Lorna K; Craig, Jonathan C; Willis, Narelle S; Tovey, David, and Webster, Angela C. How to write a cochrane systematic review. *Nephrology*, 15(6):617–624, 2010.

Hennon, Christopher C; Knapp, Kenneth R; Schreck III, Carl J; Stevens, Scott E; Kossin, James P; Thorne, Peter W; Hennon, Paula A; Kruk, Michael C; Rennie, Jared; Gadéa, Jean-Maurice, and others, . Cyclone center: can citizen scientists improve tropical cyclone intensity records? *Bulletin of the American Meteorological Society*, 96(4):591–607, 2015.

Hirth, Matthias; Hoßfeld, Tobias, and Tran-Gia, Phuoc. Cost-optimal validation mechanisms and cheat-detection for crowdsourcing platforms. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, pages 316–321. IEEE, 2011.

Hirth, Matthias; Hoßfeld, Tobias, and Tran-Gia, Phuoc. Analyzing costs and accuracy of validation mechanisms for crowdsourcing platforms. *Mathematical and Computer Modelling*, 57(11):2918–2932, 2013.

Hsu, Wei-Ning and Lin, Hsuan-Tien. Active learning by learning, 2015.

Imran, Muhammad; Castillo, Carlos; Lucas, Ji; Meier, Patrick, and Vieweg, Sarah. Aidr: Artificial intelligence for disaster response. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 159–162. ACM, 2014.

Imran, Muhammad; Mitra, Prasenjit, and Castillo, Carlos. Twitter as a lifeline: Human-annotated twitter corpora for nlp of crisis-related messages. *arXiv preprint arXiv:1605.05894*, 2016.

Jeff Pasternack, Dan Roth. Knowing what to believe (when you already know something). *International Conference on Computational Linguistics*, pages 877–885, 2010.

Jeff Pasternack, Dan Roth. Making better informed trust decisions with generalized fact-finding. *IJCAI International Joint Conference on Artificial Intelligence*, pages 2324–2329, 2011. ISSN 10450823. doi: 10.5591/978-1-57735-516-8/IJCAI11-387.

Jin, Ruochun; Dou, Yong; Wang, Yueqing, and Niu, Xin. Confusion graph: Detecting confusion communities in large scale image classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1980–1986, 2017a. doi: 10.24963/ijcai.2017/275. URL `https://doi.org/10.24963/ijcai.2017/275`.

Jin, Yuan; Carman, Mark; Kim, Dongwoo, and Xie, Lexing. Leveraging side information to improve label quality control in crowd-sourcing. In *Procs of Hcomp2017*. AAAI, 2017b.

Xindong WuJing Zhang, Victor S. Sheng. Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46(4):543–576, 2016. ISSN 15737462. doi: 10.1007/s10462-016-9491-9.

Joachims, Thorsten. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

Kamar, Ece; Hacker, Severin, and Horvitz, Eric. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '12, pages 467–474, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0-9817381-1-7, 978-0-9817381-1-6.

Kamar, Ece; Kapoor, Ashish, and Horvitz, Eric. Lifelong learning for acquiring the wisdom of the crowd. In *JCAI*, 2013.

Karger, David R; Oh, Sewoong, and Shah, Devavrat. Budget-optimal crowdsourcing using low-rank matrix approximations. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pages 284–291. IEEE, 2011a.

Karger, David R; Oh, Sewoong, and Shah, Devavrat. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*, pages 1953–1961, 2011b.

Khalil, Elias B.; Dai, Hanjun; Zhang, Yuyu; Dilkina, Bistra, and Song, Le. Learning combinatorial optimization algorithms over graphs. In Guyon, Isabelle; von Luxburg, Ulrike; Bengio, Samy; Wallach, Hanna M.; Fergus, Rob; Vishwanathan, S. V. N., and Garnett, Roman, editors, *NIPS 2017*, 2017.

Khan, Khalid S; Kunz, Regina; Kleijnen, Jos, and Antes, Gerd. Five steps to conducting a systematic review. *Journal of the Royal Society of Medicine*, 96(3):118–121, 2003.

Kipf, Thomas N. and Welling, Max. Semi-supervised classification with graph convolutional networks. In *ICLR 2017*, 2017.

Kleinberg, Jon M. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999. ISSN 0004-5411.

Kool, Wouter; van Hoof, Herke, and Welling, Max. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

Krivosheev, Evgeny; Caforio, Valentina; Benatallah, Boualem, and Casati, Fabio. Crowdsourcing paper screening in systematic literature reviews. In *Procs of Hcomp2017*. AAAI, 2017.

Krivosheev, Evgeny; Baez, Marcos; Benatallah, Boualem, and Casati, Fabio. Combining crowd and machines for multi-predicate item screening. 2018a.

Krivosheev, Evgeny; Benatallah, Boualem, and Casati, Fabio. Crowd-based multi-predicate screening of papers in literature reviews. In *Proceedings of WWW2018*. International World Wide Web Conferences Steering Committee, 2018b.

Krivosheev, Evgeny; Harandizadeh, Bahareh; Casati, Fabio, and Benatallah, Boualem. Crowd-machine collaboration for item screening. In *Companion Proceedings of the The Web Conference 2018*, pages 95–96, 2018c.

Ktena, Sofia Ira; Parisot, Sarah; Ferrante, Enzo; Rajchl, Martin; Lee, Matthew C. H.; Glocker, Ben, and Rueckert, Daniel. Distance metric learning using graph convolutional networks: Application to functional brain networks. *CoRR*, abs/1703.02161, 2017. URL `http://arxiv.org/abs/1703.02161`.

Kulesza, Todd; Burnett, Margaret; Wong, Weng-Keen, and Stumpf, Simone. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, pages 126–137, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3306-1. doi: 10.1145/2678025.2701399. URL `http://doi.acm.org/10.1145/2678025.2701399`.

Lan, Doren; Reed, Katherine; Shin, Austin, and Trushkowsky, Beth. Dynamic filter: Adaptive query processing with the crowd. In *Procs of Hcomp2017*. AAAI, 2017.

Lattimore, Tor and Szepesvar, Csaba. *Bandit Algorithms (to appear)*. Cambridge University Press, 2019. Available at http://banditalgs.com.

Law, Edith; Gajos, Krzysztof Z.; Wiggins, Andrea; Gray, Mary L., and Williams, Alex. Crowdsourcing as a tool for research: Implications of uncertainty. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, pages 1544–1561, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4335-0. doi: 10.1145/2998181.2998197. URL `http://doi.acm.org/10.1145/2998181.2998197`.

Lehmann, Jens; Isele, Robert; Jakob, Max; Jentzsch, Anja; Kontokostas, Dimitris; Mendes, Pablo N.; Hellmann, Sebastian; Morsey, Mohamed; van Kleef, Patrick; Auer, Sören, and Bizer, Christian. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

Lewis, David D. and Gale, William A. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X. URL `http://dl.acm.org/citation.cfm?id=188490.188495`.

Li, Jiyi; Baba, Yukino, and Kashima, Hisashi. Incorporating worker similarity for label aggregation in crowdsourcing. In Kůrková, Věra; Manolopoulos, Yannis; Hammer, Barbara; Iliadis, Lazaros, and Maglogiannis, Ilias, editors, *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 596–606, Cham, 2018. Springer International Publishing.

Li, Qi; Li, Yaliang; Gao, Jing; Su, Lu; Zhao, Bo; Demirbas, Murat; Fan, Wei, and Han, Jiawei. A confidence-aware approach for truth discovery on long-tail data. *VLDB*, 8(4):425–436, December 2014a.

Li, Qi; Li, Yaliang; Gao, Jing; Zhao, Bo; Fan, Wei, and Han, Jiawei. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *SIGMOD*, pages 1187–1198, 2014b.

Li, Yaliang; Gao, Jing; Meng, Chuishi; Li, Qi; Su, Lu; Zhao, Bo; Fan, Wei, and Han, Jiawei. A survey on truth discovery. *SIGKDD Explor. Newsl.*, 17(2):1–16, 2016.

Lin, Christopher H.; Mausam, , and Weld, Daniel S. Re-active learning: Active learning with relabeling. In *AAAI*, 2016.

Lin, Hongzhou and Jegelka, Stefanie. Resnet with one-neuron hidden layers is a universal approximator. In *NeurIPS 2018*, 2018.

Lintott, Chris J; Schawinski, Kevin; Slosar, Anže; Land, Kate; Bamford, Steven; Thomas, Daniel; Raddick, M Jordan; Nichol, Robert C; Szalay, Alex; Andreescu, Dan, and others, . Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 389(3):1179–1189, 2008.

Liu, Bang; Zhang, Ting; Niu, Di; Lin, Jinghong; Lai, Kunfeng, and Xu, Yu. Matching long text documents via graph convolutional networks. 02 2018.

Liu, Chao and Wang, Yi Min. Truelabel + confusions: A spectrum of probabilistic models in analyzing multiple ratings. In *Procs of ICML2012*. ICML, 2012.

Liu, Qiang; Ihler, Alexander T, and Steyvers, Mark. Scoring workers in crowdsourcing: How many control questions are enough? In *Advances in Neural Information Processing Systems*, pages 1914–1922, 2013.

Logeswaran, Lajanugen; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina; Devlin, Jacob, and Lee, Honglak. Zero-shot entity linking by reading entity descriptions. *CoRR*, abs/1906.07348, 2019. URL `http://arxiv.org/abs/1906.07348`.

Loster, Michael; Zuo, Zhe; Naumann, Felix; Maspfuhl, Oliver, and Thomas, Dirk. Improving company recognition from unstructured text by using dictionaries. In *EDBT 2017*, pages 610–619, 2017.

Ma, Guixiang; Ahmed, Nesreen K.; Willke, Theodore L.; Sengupta, Dipanjan; Cole, Michael W.; Turk-Browne, Nicholas B., and Yu, Philip S. Similarity learning with higher-order proximity for brain network analysis. *CoRR*, abs/1811.02662, 2018. URL `http://arxiv.org/abs/1811.02662`.

Manam, V. K. Chaithanya and Quinn, Alexander J. Wingit: Efficient refinement of unclear task instructions. In *Procs of HComp2018*. AAAI Publications, 2018.

Mateen, FJ; Oh, J; Tergas, AI; Bhayani, NH, and Kamdar, BB. Titles versus titles and abstracts for initial screening of articles for systematic reviews. *Clin Epidemiol.*, 5(1), 2013.

Mikolov, Tomas; Sutskever, Ilya; Chen, Kai; Corrado, Gregory S., and Dean, Jeffrey. Distributed representations of words and phrases and their compositionality. In Burges, Christopher J. C.; Bottou, Léon; Ghahramani, Zoubin, and Weinberger, Kilian Q., editors, *NIPS 2013*, pages 3111–3119, 2013.

Moher, David; Cook, Deborah J; Eastwood, Susan; Olkin, Ingram; Rennie, Drummond, and Stroup, Donna F. Improving the quality of reports of meta-analyses of randomised controlled trials: the {QUOROM} statement. *The Lancet*, 354(9193):1896 – 1900, 1999. ISSN 0140-6736. doi: http://doi.org/10.1016/S0140-6736(99)04149-5. URL `http://www.sciencedirect.com/science/article/pii/S0140673699041495`.

Moher, David; Liberati, Alessandro; Tetzlaff, Jennifer; Altman, Douglas G; Group, Prisma, and others, . Preferred reporting items for systematic reviews and meta-analyses: the prisma statement. *PLoS med*, 6(7):e1000097, 2009.

Monti, Federico; Boscaini, Davide; Masci, Jonathan; Rodolà, Emanuele; Svoboda, Jan, and Bronstein, Michael M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5425–5434, 2017.

Mortensen, Michael L.; Adam, Gaelen P.; Trikalinos, Thomas A.; Kraska, Tim, and Wallace, Byron C. An exploration of crowdsourcing citation screening for systematic reviews. *Research Synthesis Methods*, 2016. ISSN 1759-2887. RSM-02-2016-0006.R4.

Mozafari, Barzan; Sarkar, Purna; Franklin, Michael; Jordan, Michael, and Madden, Samuel. Scaling up crowd-sourcing to very large datasets: a case for active learning. *Proceedings of the VLDB Endowment*, 8(2):125–136, 2014.

Mudgal, Sidharth; Li, Han; Rekatsinas, Theodoros; Doan, AnHai; Park, Youngchoon; Krishnan, Ganesh; Deep, Rohit; Arcaute, Esteban, and Raghavendra, Vijay. Deep learning for entity matching: A design space exploration. In *SIGMOD '18*, pages 19–34, 2018.

Ng, Lauren; Pitt, Veronica; Huckvale, Kit; Clavisi, Ornella; Turner, Tari; Gruen, Russell, and Elliott, Julian H. Title and Abstract Screening and Evaluation in Systematic Reviews (TASER): a pilot randomised controlled trial of title and abstract screening by medical students. *Systematic reviews*, 3(1):121, 2014. ISSN 2046-4053 (Electronic). doi: 10.1186/2046-4053-3-121.

Nguyen, An T; Wallace, Byron C, and Lease, Matthew. Combining Crowd and Expert Labels using Decision Theoretic Active Learning. *Proceedings of the 3rd AAAI Conference on Human Computation (HCOMP)*, pages 120–129, 2015a.

Nguyen, An Thanh; Halpern, Matthew; Wallace, Byron C., and Lease, Matthew. Probabilistic modeling for crowdsourcing partially-subjective ratings. In *Procs of HComp2015*. AAAI Publications, 2015b.

Nushi, Besmira; Singla, Adish; Gruenheid, Anja; Zamanian, Erfan; Krause, Andreas, and Kossmann, Donald. Crowd access path optimization: Diversity matters. In *HCOMP*, 2015.

Parameswaran, Aditya; Garcia-Molina, Hector; Park, Hyunjung; Polyzotis, Neoklis; Ramesh, Aditya, and Widom, Jennifer. Crowdscreen: Algorithms for filtering data with humans. In *Proceedings of ACM SIGMOD*. ACM, 2012.

Parameswaran, Aditya; Boyd, Stephen; Garcia-Molina, Hector; Gupta, Ashish; Polyzotis, Neoklis, and Widom, Jennifer. Optimal crowd-powered rating and filtering algorithms. In *Proceedings of VLDB*. VLDB Endowment, 2014.

Park, Hyunjung and Widom, Jennifer. Query optimization over crowdsourced data. *Proc. VLDB Endow.*, 6(10): 781–792, August 2013. ISSN 2150-8097. doi: 10.14778/2536206.2536207. URL `http://dx.doi.org/10.14778/2536206.2536207`.

Pasternack, J and Roth, D. Making Better Informed Trust Decisions with Generalized Fact-Finding. In *IJCAI*, 2011.

Pennington, Jeffrey; Socher, Richard, and Manning, Christopher D. Glove: Global vectors for word representation. In *EMNLP*, 2014.

Pershina, Maria; Yakout, Mohamed, and Chakrabarti, Kaushik. Holistic entity matching across knowledge graphs. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 1585–1590. IEEE, 2015.

Pochampally, Ravali; Das Sarma, Anish; Dong, Xin Luna; Meliou, Alexandra, and Srivastava, Divesh. Fusing data with correlations. In *SIGMOD*, pages 433–444. ACM Press, 2014.

Mahimarnab HajraPratap Chandra Sen, ansMitadru Ghosh. *Supervised Classification Algorithms in Machine Learning: A Survey and Review*, pages 99–111. 01 2020. ISBN 978-981-13-7402-9. doi: 10.1007/978-981-13-7403-6_11.

Ramírez, Jorge; Baez, Marcos; Casati, Fabio, and Benatallah, Boualem. Understanding the impact of text highlighting in crowdsourcing tasks. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pages 144–152, 2019.

Ranard, Benjamin L.; Ha, Yoonhee P.; Meisel, Zachary F.; Asch, David A.; Hill, Shawndra S.; Becker, Lance B.; Seymour, Anne K., and Merchant, Raina M. Crowdsourcingharnessing the masses to advance health and medicine, a systematic review. *Journal of General Internal Medicine*, 29(1), 2014.

Ribeiro, Marco Tulio; Singh, Sameer, and Guestrin, Carlos. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939778. URL `http://doi.acm.org/10.1145/2939672.2939778`.

Mani SrivastavaRobin Ouyang, Alice Toniolo Timothy J. Norman. Truth discovery in crowdsourced detection of spatial events. *IEEE Transactions on Knowledge and Data Engineering*, 28(4):1047–1060, 2016. ISSN 10414347. doi: 10.1109/TKDE.2015.2504928.

Rodriguez, Carlos; Daniel, Florian, and Casati, Fabio. Crowd-based mining of reusable process model patterns. In Sadiq, Shazia; Soffer, Pnina, and Völzer, Hagen, editors, *Business Process Management*, pages 51–66. Springer International Publishing, 2014.

Rokach, Lior. Ensemble-based classifiers. *Artif. Intell. Rev.*, 33(1-2):1–39, February 2010. ISSN 0269-2821. doi: 10.1007/s10462-009-9124-7. URL `http://dx.doi.org/10.1007/s10462-009-9124-7`.

Rousseeuw, Peter. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. 1987.

Russakovsky, Olga; Deng, Jia; Su, Hao; Krause, Jonathan; Satheesh, Sanjeev; Ma, Sean; Huang, Zhiheng; Karpathy, Andrej; Khosla, Aditya; Bernstein, Michael; Berg, Alexander C., and Fei-Fei, Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec 2015. ISSN 1573-1405. doi: 10.1007/s11263-015-0816-y. URL `https://doi.org/10.1007/s11263-015-0816-y`.

Sampson, Margaret; Shojania, Kaveh G; Garritty, Chantelle; Horsley, Tanya; Ocampo, Mary, and Moher, David. Systematic reviews can be produced and published faster. *Journal of clinical epidemiology*, 61(6):531–536, 2008. ISSN 08954356.

Yukino BabaSatoshi Oyama, Yuko Sakurai Hisashi Kashima. Accurate integration of crowdsourced labels using workers self-reported confidence scores. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 2554–2560, 2013.

Scarselli, Franco; Gori, Marco; Tsoi, Ah Chung; Hagenbuchner, Markus, and Monfardini, Gabriele. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605. URL `https://doi.org/10.1109/TNN.2008.2005605`.

Schlichtkrull, Michael Sejr; Kipf, Thomas N.; Bloem, Peter; van den Berg, Rianne; Titov, Ivan, and Welling, Max. Modeling relational data with graph convolutional networks. In Gangemi, Aldo; Navigli, Roberto; Vidal, Maria-Esther; Hitzler, Pascal; Troncy, Raphaël; Hollink, Laura; Tordai, Anna, and Alam, Mehwish, editors, *ESWC 2018*, 2018.

Schroff, Florian; Kalenichenko, Dmitry, and Philbin, James. Facenet: A unified embedding for face recognition and clustering. *CVPR*, pages 815–823, 2015.

Selsam, Daniel; Lamm, Matthew; Bünz, Benedikt; Liang, Percy; de Moura, Leonardo, and Dill, David L. Learning a SAT solver from single-bit supervision. *CoRR*, abs/1802.03685, 2018. URL `http://arxiv.org/abs/1802.03685`.

Shen, Yanyao; Yun, Hyokun; Lipton, Zachary; Kronrod, Yakov, and Anandkumar, Animashree. Deep active learning for named entity recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 252–256. Association for Computational Linguistics, 2017. doi: 10.18653/v1/W17-2630. URL `http://aclweb.org/anthology/W17-2630`.

Shervashidze, Nino; Schweitzer, Pascal; van Leeuwen, Erik Jan; Mehlhorn, Kurt, and Borgwardt, Karsten M. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.*, 12:2539–2561, 2011.

Joel MathewShobeir Fakhraei, Jose Luis Ambite. Nseen: Neural semantic embedding for entity normalization. *PKDD*, 2019.

Simpson, Robert; Page, Kevin R., and De Roure, David. Zooniverse: Observing the world's largest citizen science platform. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 1049–1054, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2745-9. doi: 10.1145/2567948.2579215. URL `http://doi.acm.org/10.1145/2567948.2579215`.

Smyth, Padhraic; Fayyad, Usama; Burl, Michael; Perona, Pietro, and Baldi, Pierre. Inferring ground truth from subjective labelling of venus images. *Advances in neural information processing systems*, 7:1085–1092, 1995.

Steinberg, Earl; Greenfield, Sheldon; Wolman, Dianne Miller; Mancher, Michelle; Graham, Robin, and others, . *Clinical practice guidelines we can trust*. National Academies Press, 2011.

Sukhobok, Dina; Nikolov, Nikolay; Pultier, Antoine; Ye, Xianglin; Berre, Arne J.; Moynihan, Rick; Roberts, Bill; Elvesæter, Brian; Nivethika, Mahasivam, and Roman, Dumitru. Tabular data cleaning and linked data generation with grafterizer. In Sack, Harald; Rizzo, Giuseppe; Steinmetz, Nadine; Mladenic, Dunja; Auer, Sören, and Lange, Christoph, editors, *The Semantic Web - ESWC 2016 Satellite Events, Revised Selected Papers*, pages 134–139, 2016.

Sun, Yalin; Cheng, Pengxiang; Wang, Shengwei; Lyu, Hao; Lease, Matthew; Marshall, Iain, and Wallace, Byron C. Crowdsourcing Information Extraction for Biomedical Systematic Reviews. In *4th AAAI Conference on Human Computation and Crowdsourcing (HCOMP): Works-in-Progress Track*, 2016. URL `http://arxiv.org/abs/1609.01017`. 3 pages. arXiv:1609.01017.

Swanson, Alexandra; Kosmala, Margaret; Lintott, Chris; Simpson, Robert; Smith, Arfon, and Packer, Craig. Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna. *Scientific data*, 2:150026, 2015.

Takwoingi, Yemisi; Hopewell, Sally; Tovey, David, and Sutton, Alex J. A multicomponent decision tool for prioritising the updating of systematic reviews. *Bmj*, 7191(December):1–8, 2013. ISSN 1756-1833. doi: 10.1136/bmj.f7191. URL `http://www.bmj.com/content/bmj/347/bmj.f7191.full.pdf`.

Tinati, Ramine; Kleek, Max Van; Simperl, Elena; Luczak-Roesch, Markus; Simpson, Robert, and Shadbolt, Nigel. Designing for citizen data analysis: A cross-sectional case study of a multi-domain citizen science platform. In *Procs of CHI 2015*. ACM Press, 2015.

Trivedi, Rakshit; Sisman, Bunyamin; Dong, Xin Luna; Faloutsos, Christos; Ma, Jun, and Zha, Hongyuan. Linknbed: Multi-graph representation learning with entity linkage. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 252–262, 2018.

van der Maaten, Laurens and Hinton, Geoffrey. Visualizing data using t-sne, 2008.

Van Horn, G.; Branson, S.; Farrell, R.; Haber, S.; Barry, J.; Ipeirotis, P.; Perona, P., and Belongie, S. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 595–604, June 2015. doi: 10.1109/CVPR.2015.7298658.

Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N.; Kaiser, Lukasz, and Polosukhin, Illia. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 6000–6010, 2017.

Vaughan, Jennifer Wortman. Making better use of the crowd: How crowdsourcing can advance machine learning research. Survey and position paper, Microsoft Research, 2017. Available at http://www.jennwv.com/projects/crowdtutorial.html.

Velickovic, Petar; Cucurull, Guillem; Casanova, Arantxa; Romero, Adriana; Liò, Pietro, and Bengio, Yoshua. Graph attention networks. In *ICLR*, 2018.

Veronese, Nicola; Facchini, Silvia; Stubbs, Brendon; Luchini, Claudio; Solmi, Marco; Manzato, Enzo; Sergi, Giuseppe; Maggi, Stefania; Cosco, Theodore, and Fontana, Luigi. Weight loss is associated with improvements in cognitive function among overweight and obese people: A systematic review and meta-analysis. *Neuroscience & Biobehavioral Reviews*, 72:87–94, 2017.

Vesdapunt, Norases; Bellare, Kedar, and Dalvi, Nilesh. Crowdsourcing algorithms for entity resolution. In *Proceedings of VLDB*. VLDB Endowment, 2014.

Vinayak, Ramya Korlakai and Hassibi, Babak. Crowdsourced clustering: Querying edges vs triangles. In *Procs of Nips 2016*, 2016.

Wallace, Byron C; Dahabreh1, Issa J; Schmid, Christopher H; Lau1, Joseph, and Trikalinos, Thomas A. Modernizing the systematic review process to inform comparative effectiveness: tools and methods. *J. Compar. Effect. Res.*, 2(3), 2013.

Wallace, Byron C; Noel-Storr, A; Marshall, IJ; Cohen, AM; Smalheiser, NR, and Thomas, J. Identifying reports of randomized controlled trials (rcts) via a hybrid machine learning and crowdsourcing approach. *J Am Med Inform Assoc*, 2017a.

Wallace, Byron C; Noel-Storr, A; Marshall, IJ; Cohen, AM; Smalheiser, NR, and Thomas, J. Identifying reports of randomized controlled trials (rcts) via a hybrid machine learning and crowdsourcing approach. *J Am Med Inform Assoc*, 2017b.

Wallace, Byron C; Noel-Storr, Anna; Marshall, Iain J; Cohen, Aaron M; Smalheiser, Neil R, and Thomas, James. Identifying reports of randomized controlled trials (rcts) via a hybrid machine learning and crowdsourcing approach. *J Am Med Inform Assoc*, 2017c.

Wang, Chong; Zhang, Xue, and Lan, Xipeng. How to train triplet networks with 100k identities? *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 1907–1915, 2017.

Wang, Jiangtao; Wang, Yasha, and Lv, Qin. Crowd-assisted machine learning: Current issues and future directions. *Computer*, 52(1):46–53, January 2019. ISSN 0018-9162. doi: 10.1109/MC.2018.2890174. URL `https://doi.org/10.1109/MC.2018.2890174`.

Wang, Jiannan; Kraska, Tim; Franklin, Michael J., and Feng, Jianhua. Crowder: Crowdsourcing entity resolution. In *Proceedings of VLDB*. VLDB Endowment, 2012.

Wang, Xianzhi; Sheng, Quan Z; Fang, Xiu Susie, and Yao, Lina. An Integrated Bayesian Approach for Effective Multi-Truth Discovery. In *CIKM*, 2015.

Wang, Xiaolong; Girshick, Ross B.; Gupta, Abhinav, and He, Kaiming. Non-local neural networks. In *IEEE CVPR 2018*, pages 7794–7803, 2018.

Weisfeiler, Boris and Lehman, Andrei A. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

Weiss, Michael. Crowdsourcing literature reviews in new domains. *Technology Innovation Management Review*, 6 (2):5–14, 2016.

Whitehill, Jacob; Wu, Ting-fan; Bergsma, Jacob; Movellan, Javier R, and Ruvolo, Paul L. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, pages 2035–2043, 2009.

Wright, Rick W; Brand, Richard A; Dunn, Warren, and Spindler, Kurt P. How to write a systematic review. *Clinical orthopaedics and related research*, 455:23–29, 2007.

Xirogiannopoulos, Konstantinos and Deshpande, Amol. Extracting and analyzing hidden graphs from relational databases. In *SIGMOD*, 2017.

Xu, Keyulu; Li, Chengtao; Tian, Yonglong; Sonobe, Tomohiro; Kawarabayashi, Ken-ichi, and Jegelka, Stefanie. Representation learning on graphs with jumping knowledge networks. In *Proceedings of ICML 2018*, 2018.

Xu, Keyulu; Hu, Weihua; Leskovec, Jure, and Jegelka, Stefanie. How powerful are graph neural networks? *ICLR 2019*, 2019.

Yang, Jie; Redi, Judith; DeMartini, Gianluca, and Bozzon, Alessandro. Modeling task complexity in crowdsourcing. In *Proceedings of The Fourth AAAI Conference on Human Computation and Crowdsourcing (HCOMP 2016)*, pages 249–258. AAAI, 2016.

Yang, Jie; Drake, Thomas; Damianou, Andreas, and Maarek, Yoelle. Leveraging crowdsourcing data for deep active learning an application: Learning intents in alexa. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 23–32. International World Wide Web Conferences Steering Committee, 2018.

Ying, Zhitao; You, Jiaxuan; Morris, Christopher; Ren, Xiang; Hamilton, William L., and Leskovec, Jure. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS 2018*, pages 4805–4815, 2018.

You, Jiaxuan; Liu, Bowen; Ying, Zhitao; Pande, Vijay, and Leskovec, Jure. Graph convolutional policy network for goal-directed molecular graph generation. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N., and Garnett, R., editors, *NIPS 18*. 2018a.

You, Jiaxuan; Ying, Rex; Ren, Xiang; Hamilton, William L., and Leskovec, Jure. Graphrnn: A deep generative model for graphs. *CoRR*, abs/1802.08773, 2018b. URL `http://arxiv.org/abs/1802.08773`.

Zhang, Muhan; Cui, Zhicheng; Neumann, Marion, and Chen, Yixin. An end-to-end deep learning architecture for graph classification. In *Proceedings of AAAI-18, IAAI-18, and EAAI-18*, 2018.

Zhang, Wen; Shu, Kai; Liu, Huan, and Wang, Yalin. Graph neural networks for user identity linkage. *CoRR*, abs/1903.02174, 2019. URL `http://arxiv.org/abs/1903.02174`.

Zhao, Bo; Rubinstein, Benjamin I. P.; Gemmell, Jim, and Han, Jiawei. A bayesian approach to discovering truth from conflicting sources for data integration. *VLDB*, pages 550–561, 2012.

Zheng, Yudian; Li, Guoliang; Li, Yuanbing; Shan, Caihua, and Cheng, Reynold. Truth inference in crowdsourcing: Is the problem solved? *Proc. VLDB Endow.*, 10(5):541–552, January 2017. ISSN 2150-8097. doi: 10.14778/3055540.3055547. URL `https://doi.org/10.14778/3055540.3055547`.