UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

TOWARDS RELIABLE HYBRID HUMAN-MACHINE CLASSIFIERS

Burcu Sayin Günel

Advisor

Prof. Fabio Casati

Università degli Studi di Trento

Co-Advisor

Prof. Andrea Passerini

Università degli Studi di Trento

September 2022

Abstract

In this thesis, we focus on building reliable hybrid human-machine classifiers to be deployed in cost-sensitive classification tasks. The objective is to assess ML quality in hybrid classification contexts and design the appropriate metrics, thereby knowing whether we can trust the model predictions and identifying the subset of items on which the model is well-calibrated and trustworthy. We start by discussing the key concepts, research questions, challenges, and architecture to design and implement an effective hybrid classification service. We then present a deeper investigation of each service component along with our solutions and results. We mainly contribute to cost-sensitive hybrid classification, selective classification, model calibration, and active learning. We highlight the importance of model calibration in hybrid classification services and propose novel approaches to improve the calibration of human-machine classifiers. In addition, we argue that the current accuracy-based metrics are misaligned with the actual value of machine learning models and propose a novel metric "value". We further test the performance of SOTA machine learning models in NLP tasks with a cost-sensitive hybrid classification context. We show that the performance of the SOTA models in cost-sensitive tasks significantly drops when we evaluate them according to value rather than accuracy. Finally, we investigate the quality of hybrid classifiers in the active learning scenarios. We review the existing active learning strategies, evaluate their effectiveness, and propose a novel value-aware active learning strategy to improve the performance of selective classifiers in the active learning of cost-sensitive tasks.

Keywords
machine learning, active learning, classification, hybrid intelligence

Acknowledgements

I would like to express my sincere gratitude to my advisor, Prof. Fabio Casati, who gave me this fantastic opportunity to undertake my Ph.D. studies at the Department of Information Engineering and Computer Science, University of Trento. For three years and a half, I learned a lot from him. Thank you so much for guiding me on this rocky road, sharing your experience, helping me to solve challenging problems, and making things easier.

I owe a big thank you to my co-advisor, Prof. Andrea Passerini; I am grateful to you for supporting me in a critical period of my doctoral studies. Thank you so much for taking the time to guide me, for giving me the chance to be part of your research group, and to work with amazing people. I enjoyed every moment of my work in the SML group; thank you all for filling my Ph.D. with unforgettable memories and joy.

My sincere thanks also go to Prof. Jie Yang; I am grateful to you for your collaboration and great feedback on the different projects I carried out during my Ph.D. It was a great opportunity for me to collaborate with the KAPPA Lab at the Delft University of Technology. I have unforgettable memories from our online meetings; a big thank you to all!

To my husband, this period would have been much more difficult for me without your support and patience. Thank you for always being there and cheering for me. To my sister and brother, your presence is indescribable happiness for me; thank you so much for always showing your support and love. To my friends, I can not express my gratitude for your unconditional support. Thank you for coping with the distance and giving me the strength to keep going.

My father and mother, this Ph.D. is for you! I am grateful to you for your unconditional support throughout my life, endless trust in every situation, and always making me feel your love. We receive the recompense of our effort together; I love you!

Trento, September 2022

Contents

Chapter 1

Introduction

Machine learning (ML) and deep learning (DL) have become a compelling research domain where the researchers aim to develop intelligent systems that can replace human effort in many complex tasks, from speech recognition to medical diagnosis and autonomous driving. Although we witnessed that artificial intelligence (AI) can beat humans in many domains[1] (e.g. visual recognition, playing games, and even detecting legal issues in law), this can not be generalized to every case.

When we need to build safety-critical systems, it becomes vital to know whether we can trust the model or not. Consider the case of self-driving cars in which the ML model should work in a very complex environment where unexpected things may happen at any time. If the autonomous driving module is not trustworthy enough, we may end with fatal accidents. Let's think of the company Tesla which made appreciable progress in the field of self-driving cars. Its auto-pilot system has made breakthroughs in recent years, but according to the New York Post news on August 16, 2021[2], "there have been 11 accidents in which Tesla was in autopilot mode and hit the emergency vehicles". So, the question remains whether DL in its current state will be enough to overcome all the challenges of self-driving. Object detection, velocity, and range estimation

[1] https://tinyurl.com/AI-beats-human

[2] https://tinyurl.com/Tesla-crashes

play a big part in driving, but human vision also performs many other complex functions. DL models also struggle with making causal inferences, which can be a huge barrier when the models face new situations they have not seen before. While Tesla created a large and diverse dataset, open roads are very complex environments where new and unpredictable things can always happen. This observation motivates the need for humans to make critical decisions in AI.

Humans are almost always involved in the ML loops when we deploy the ML models into real enterprise scenarios, either to provide ground truth labels as experts or to give feedback about the machine predictions. Human feedback becomes more critical if the use case requires some safety-critical decisions and if an erroneous prediction would cause a high cost. When this is the case, it becomes crucial to know if we can trust the model.

As Socrates said: "The only true wisdom is in knowing that you know nothing"; there is always more information we can learn, and there is always a possibility to make wrong inferences even as a human. Considering that some tasks may be very challenging even for humans, we can accept that machines are not always reliable. In order to build robust AI systems, we need ML models that are aware of what they do not know and give alerts in case of being unsure about critical decisions. However, a principal challenge of DL models today is being over-confident about their predictions and making a lot of high confidence errors. Thus, we can not just rely on the confidence of machines, and we need to build some rejection mechanisms to decide when to reject machine decisions and involve human judgments to obtain more trustable predictions.

Last but not least, once we include a reject option in ML processes, we need to rethink and re-evaluate how we can measure the quality of ML models. If the ML model has a reject option, then accuracy is not the metric to use; we need novel metrics to assess the real value of ML models in selective model settings. Moreover, model calibration or the accuracy of confidence estimates becomes at least as important as accuracy.

1.1 The Context

This thesis focuses on the hybrid human-machine classification contexts where we solve the classification tasks via combined contribution of humans and machines, and the main objective is to build reliable hybrid classification services.

1.2 The Problem

Figure 1.1 shows a hybrid classification scenario where the system takes a set of unclassified documents as input and uses the combined contribution of humans and machines to reach the classified documents.

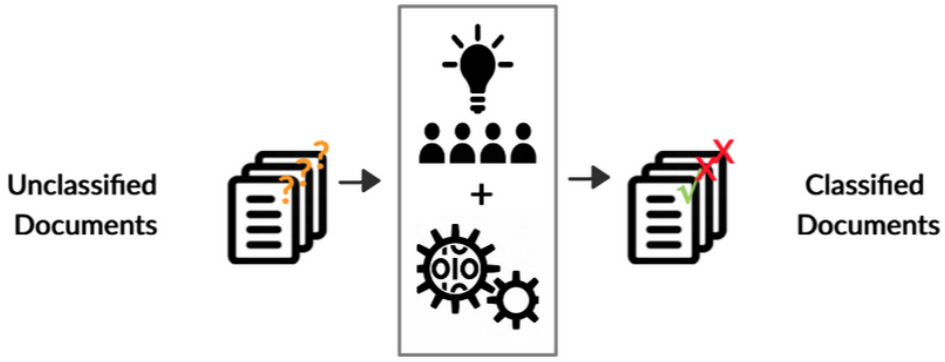


Figure 1.1: Hybrid classification

The black-box model of a hybrid classifier includes an ML model that is either pre-trained or fine-tuned on the given task and a group of humans. From a high-level perspective, the model outputs an inference for each document associated with confidence scores, and then the system accepts some of the machine inferences (considering the confidence scores) while leaving the rest for human verification. Here the assumption is that the model scores are trustable, and the humans are experts who provide the ground truth labels in every case. However, these assumptions are not realistic; (i) human labels can be noisy, (ii) the model may not be well-calibrated to provide realistic confidence scores, and (iii) even

if the model outputs are trustable, it is highly challenging to decide a confidence threshold used to accept or reject a machine inference. Thus, the problem is how to design a reliable hybrid human-machine classification service that can output robust inferences given any cost-specific use case, ML model and humans.

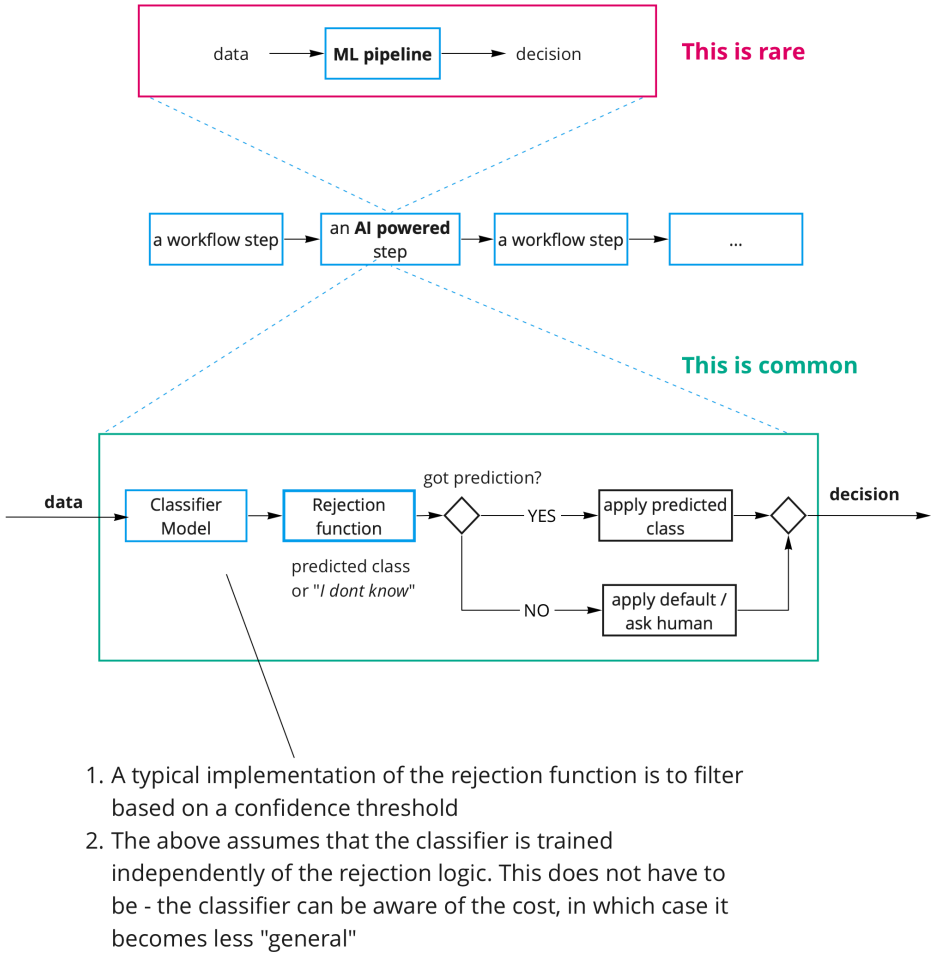


Figure 1.2: Typical implementation of ML models into an enterprise workflow (taken from our paper [122]).

Figure 1.2 shows the typical deployment of AI solutions in an end-to-end enterprise workflow. First, we either reuse or fine-tune a pre-trained model or use it in combination with some task-specific model. Either way, we have some ML classifier that, given input data, produces a predicted class and confidence (or distribution of predicted class with confidence scores). Today, there is almost

always filtering in model deployment based on whether the model confidence is higher than some threshold. If so, the prediction is applied else: a default path comes up. This is true from autonomous driving or reading Xrays to detecting intents in a chatbot to automatically routing requests, and in nearly any use case we can imagine. The point is that this is not the exception; this is the rule. That is almost always how we use ML in practice in any production scenario, and the notion of having some humans in the loop in the prediction cycle is not the exception. There are always many humans in many ML loops. At a minimum, some humans enjoy the benefits of prediction or suffer from consequences. If we accept this notion that models are almost always implemented as *selective classifiers* or *selective models*, then how we choose one model other than the other changes.

What we observed from our research is that, first of all, the use case drives the value of ML models. Given the use case, we have a certain cost for (i) rejecting a prediction, (ii) applying a correct prediction, and (iii) applying a wrong prediction. These value parameters help us to decide where we set our threshold to accept or reject the model prediction. And, because we have a confidence threshold, model accuracy is not the priority for us; if we have a well-calibrated model with arbitrarily bad accuracy, we can still get value from it. The better we can identify the subset of items for which our model is well-calibrated, the lower the cost for our deployment in an AI workflow. The point is to trust our model and let it tell us what it knows and what it does not know indeed.

1.3 The proposed approach

Our approach is to build a hybrid classification service that effectively combines humans and machines to output reliable predictions. Figure 1.3 (taken from our paper [120] and modified) shows how we envision such a service.

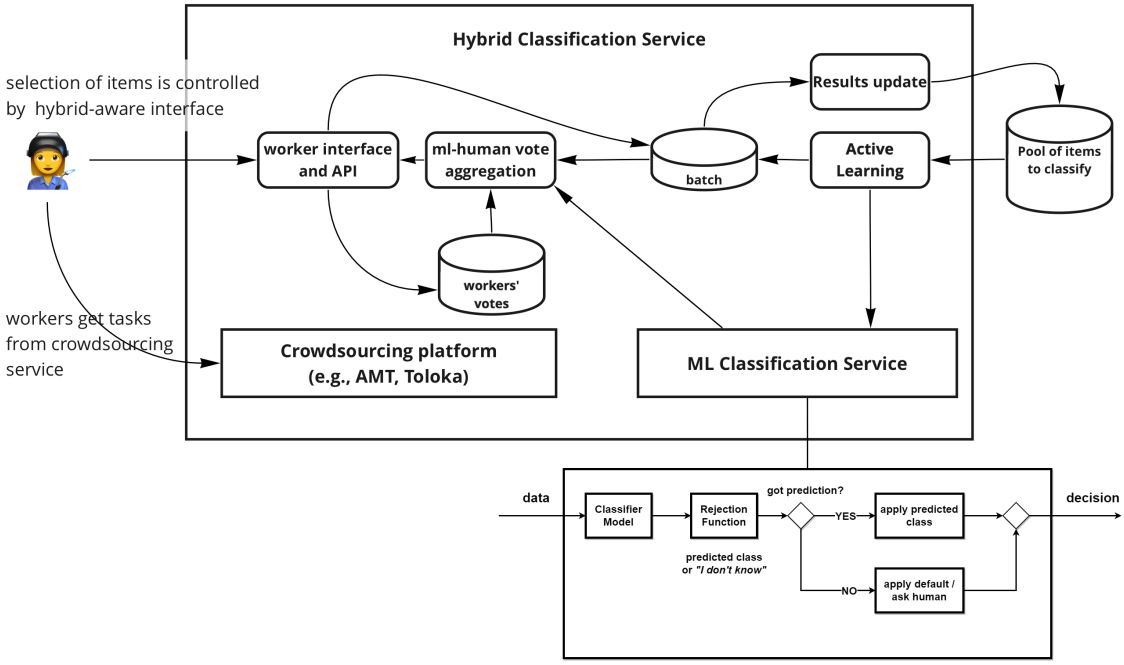


Figure 1.3: Hybrid Classification Service

A crowdsourcing platform (CP) provides basic access to users and manages payment. While in most cases, the CP is also responsible for serving the specific items to users (workers) for labeling, we need to take control and override that logic because we want hybrid classification to come into play. Specifically, we want to be able to decide for which items we ask crowd labels for, and to how many persons. We also want to dynamically stop asking if the combination of machine prediction and the votes obtained so far gives us enough confidence to make a decision. Therefore, the items are loaded externally to the CP, into an item pool (on the right in the figure).

Processing proceeds in batches. The optimal batch of items is selected via an "active learning" (AL) component that is peculiar in this case because we have two sometimes conflicting optimization objectives; asking votes to classify items in the pool vs. asking votes based on what is best for training an ML model that can then help us (this is somewhat more nuanced than traditional AL approaches, and discussed further in [77]).

Once we do have a batch of votes, we then classify items through a combination of humans and ML. As widely done in crowdsourcing, we can leverage the many existing techniques that compute workers' accuracy and aggregate votes on an item, using a variation of the Expectation-Maximization algorithm, such as Dawid-Skene [44]. However, the approach we take is to consider the classifier as a "voter", much like human workers, that is, characterized by its own "accuracy", where the accuracy (unlike workers) varies per item and corresponds to the prediction confidence on that item. Notice that in some cases, the classifier may be so confident that its vote is sufficient, and we do not need to resort to any crowd vote. Notice also that while workers' accuracy is independent of the items (we do not assume the crowdsourcing tasks ask for probabilities, as that is not commonly done and it would raise a whole set of challenges), ML classifiers typically do output a per-item probability, and we use it here.

Research questions

Many research questions should be investigated to design and implement such a hybrid classification service. In this thesis, we specifically focus on the following questions related to different components:

- *ML Classification Service*
  - **RQ1.** How to measure quality in ML, and is accuracy a good measure of quality in the hybrid classification context?
  - **RQ2.** How does the model calibration affect the performance of ML models in hybrid classification contexts? And how can we obtain well-calibrated hybrid classifiers?
  - **RQ3.** What are the proper metrics for the cost of using ML with rejection?
  - **RQ4.** How to effectively characterize ML failures?

- *Active learning*
  - **RQ5.** How to use AL in hybrid classification contexts? Are the existing AL strategies cost-effective, or do we need novel cost-aware AL methods for hybrid classification services?
- *ML-Human vote aggregation*
  - **RQ6.** How to efficiently combine crowdsourcing and machine intelligence?

1.4 Structure of the Thesis

This thesis comprises the work carried out during the 3-years Ph.D. Program in Information and Communication Technologies at the University of Trento, with a productive collaboration with Delft University of Technology. The objective is to build reliable hybrid human-machine classifiers that can be deployed in hybrid classification services. First, we investigate the effect of calibration in hybrid classification and propose novel methods to improve the model calibration. After that, we discuss the idea of rejecting machine decisions to maximize the value of ML models. Following this, we criticize that accuracy-based metrics are not good enough to assess the quality of ML models and propose a novel metric to evaluate the performance of ML models in cost-sensitive classification tasks. Then, we motivate and discuss the importance of AL in hybrid classification settings, analyze the effectiveness of the existing AL strategies, and propose alternative techniques to improve their adaptabilities to cost-sensitive hybrid classification problems. Finally, we show how the active learning of classifiers is affected by the focus on value.

Chapter 2 aims at taking the first step toward science for hybrid classification services, and particularly focuses on RQ2. We discuss key concepts, challenges,

and architectures and then focus on a central aspect; ML calibration and how it can be achieved with crowdsourced labels [120].

Chapter 3 motivates why the science of learning to reject model predictions is central to ML, and why human computation has a lead role in this effort. We investigate RQ1, RQ2, RQ3, RQ4, and RQ6 from a general perspective and discuss possible solutions [122].

Chapter 4 argues that the way we have been training and evaluating ML models so far has substantially forgotten the fact that they are applied in an organization or societal context because they provide value to people. We make a deeper investigation of RQ1 and RQ3, and show that when we take this perspective, we fundamentally change how we evaluate, select and deploy ML models - and to some extent, even what it means to "learn". Specifically, we stress that the notion of *value* plays a central role in learning and evaluating, and different models may require different learning practices and provide different values based on the application context in which they are applied. We also show how this concretely impacts how we select and embed models into human workflows based on experimental datasets [Under review, ACL 2023].

Chapter 5 mainly explores RQ5 by reviewing the existing AL approaches and presenting the experimental analysis of a set of bench-marking ones on crowdsourced datasets. We provide a comprehensive and systematic survey of the recent research on AL in the hybrid human-machine classification setting, where crowd workers contribute labels (often noisy) to either directly classify data instances or train machine learning models. We identify three categories of state-of-the-art (SOTA) AL methods according to whether and how predefined queries are employed for data sampling, namely fixed-strategy approaches, dynamic-strategy approaches, and strategy-free approaches. We then conduct an empirical study on their cost-effectiveness, showing that the performance of the existing AL approaches is affected by many factors in hybrid classification contexts, such as the noise level of data, label fusion technique

used, and the specific characteristics of the task. Finally, we discuss challenges and identify potential directions to design AL strategies for hybrid classification problems [121].

Chapter 6 explores how the active learning of selective classifiers is affected by the value, and how this affects the performance of AL strategies. In particular, we investigate if uncertainty sampling [81] has an edge over random sampling, and if simple alternative strategies can outperform both uncertainty and random sampling. We then propose a novel value-aware AL strategy that outperforms the state-of-the-art ones when the cost of incorrect predictions substantially outweighs that of abstaining [Under review, EMNLP 2022].

1.5 Contributions

This thesis contributes to cost-sensitive human-machine (hybrid) classification, selective classification, model calibration, and active learning. The contributions can be summarized as follows.

We designed the concepts and architecture of a hybrid classification service and showed that (i) the service should effectively combine the ML with crowdsourcing given the objective of the task (i.e. classify as many items as possible given a fixed budget or training the model) by controlling the classification process and dynamically determining if and how many crowd votes we ask for an item, (ii) the key metric for the service should be calibration rather than the accuracy; the ability to estimate what crowd and ML can do and which kind of items they can or cannot classify accurately becomes central to the effectiveness of the service. We also proposed novel methods to improve the calibration of human-machine classifiers in contexts where the loss function is skewed and the cost of errors is high compared to the cost of asking humans.

We discussed that current ML quality metrics are misaligned with the value and cost of ML models in hybrid classification contexts. We proposed that

we need a new set of metrics as well as a science for learning when to reject machine inferences. We highlighted the importance of identifying the subsets of items where we can trust the model.

We criticized that universal quality metrics used for ML model evaluation can lead to wrong decisions in hybrid classification context. We showed that the common approach for rejecting predictions (filtering by confidence threshold) leads to low or even negative model value if applied naively and to a significant loss of ”value” concerning what can be achieved. We proposed that this is true even when calibration methods are applied, and discussed why measures of calibration errors fail to capture the most important property of confidence scores, that is, the fact that the probability of confidence is high when predictions are correct (rather than the other way around). Finally, we showed that simple, decades-old models, especially when trained in-domain even with simple text encoders, behave very well and often better than large, complex models and provided some intuition of why that may be.

We published a literature review of AL strategies along with an experimental analysis of their characteristics and effectiveness. We reported the results of an extensive experimental evaluation, providing insights into the performance of existing AL strategies in hybrid human-machine classification contexts. We discussed the main insights from our analysis and highlighted relevant open challenges and potential future directions to address them. We further contributed a library of implementations of state-of-the-art AL strategies and a collection of benchmarking datasets for human-machine classification[3].

We investigated how active learning of selective classifiers is affected by the focus on value. We showed that the performance of the state-of-the-art active learning strategies in cost-sensitive tasks drops significantly when we evaluate them according to value rather than accuracy. Finally, we proposed a novel value-aware active learning strategy that outperforms the state-of-the-art ones

[3]https://tinyurl.com/source-code-data-results

when the cost of incorrect predictions substantially outweighs that of abstaining.

1.6 Publications

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

1. Burcu Sayin, Fabio Casati, Andrea Passerini, Jie Yang, and Xinyue Chen. *Rethinking and Recomputing the Value of ML Models*. 2023. Under Review for ACL 2023.

2. Burcu Sayin, Fabio Casati, Andrea Passerini, and Jie Yang. *Value-aware active learning*. 2022. Under Review for EMNLP 2022.

3. Shahin Sharifi, Sihang Qiu, Burcu Sayin, Agathe Balayn, Ujwal Gadiraju, Jie Yang, and Alessandro Bozzon. *Perspective: Leveraging Human Understanding for Identifying and Characterizing Image Atypicality*[4]. 2022. Under Review for CSCW 2022.

4. Burcu Sayin, Jie Yang, Andrea Passerini, and Fabio Casati. *The Science of Rejection: A Research Area for Human Computation*[5]. In The 9th AAAI Conference on Human Computation and Crowdsourcing (HCOMP 2021) [122].

5. Burcu Sayin, Evgeny Krivosheev, Jie Yang, Andrea Passerini, and Fabio Casati. *A review and experimental analysis of active learning over crowdsourced data*. Journal of Artificial Intelligence Review, October, 2021 [121].

6. Jorge Ramírez, Burcu Sayin, Marcos Baez, Fabio Casati, Luca Cernuzzi, Boualem Benatallah, and Gianluca Demartini. *On the State of Reporting in*

[4]This work is done as a part of CATS4ML challenge conducted by Google in 2021.
[5]This work received the Blue Sky Best Paper award from AAAI.

Crowdsourcing Experiments and a Checklist to Aid Current Practices[6]. In Proceedings of The ACM on Human-Computer Interaction (PACM HCI), presented at the 24th ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW 2021) [110].

7. Burcu Sayin, Evgeny Krivosheev, Jorge Ramirez, Fabio Casati, Ekaterina Taran, Veronika Malanina, and Jie Yang. *Crowd-Powered Hybrid Classification Services: Calibration is all you need.* In 28th IEEE International Conference on Web Services (ICWS 2021) [120].

8. Evgeny Krivosheev, Burcu Sayin, Alessandro Bozzon, and Zoltán Szlávik. *Active Learning from Crowd in Document Screening*. In Crowd Science Workshop at 34th Conference on Neural Information Processing Systems (NeurIPS 2020) [78].

[6]This work received the Methods Recognition award from ACM.

Chapter 2

Calibration in hybrid classification

Hybrid classification (HC) refers to the process of solving classification problems by leveraging both humans and ML [75]. Hybrid classification services have received little attention in the literature but, in our experience are by far the most common form of classification, especially in an enterprise context. Almost invariably, when ML is adopted in the enterprise as well as in domains from medical to automotive, ML attempts an inference, and a decision on whether to accept the inference or ask a human ensues, based on the *prediction confidence*.

Model calibration refers to the process of adjusting model parameters so that the prediction confidence is an accurate estimation of the probability of the prediction being correct [58]. Calibration is extremely important in all cases where failures are costly and a fallback option to delegate decisions to human exists, from self-driving cars to automated medical diagnosis and even in work automation. For example, in enterprise workflow companies, a well-calibrated model is central to the adoption of AI as it gives customers the peace of mind to know that they have a model that knows when it doesn't know. Thus, the platform can decide, for each inference, when to trust it (because it has high confidence) and when instead to route the decision problem to a person. More generally, we argue that any time asking humans is an option, that possibility needs to be factored in as a first-class citizen and design variable, when training and using

ML services, along with its cost (effort, money) and benefits (possibly, a more accurate classification, especially in cases where ML is undecided).

This chapter proposes a *crowd-powered hybrid classification service*. Our goal is to take a step in the direction of developing science for services that, automatically or semi-automatically, combine ML, crowd, and experts in a cost-efficient and effective way to solve batch (finite item pool) or online (infinite pool) classification problems.

Specifically, the classes of problems we tackle are those with the following characteristics[1]:

- We have the option of asking humans for each classification, though at a cost. "Humans" can be experts and/or can be "crowd workers", accessed through a crowdsourcing platform, such as Mechanical Turk or Toloka.
- We have access to ML classification services, at a cost per classification that we assume is negligible concerning the cost of asking humans.
- The cost of asking humans is significantly less than the "cost" of either a false positive or a false negative.

In this very common scenario, given a budget, a crowd of labelers available at a price, a cost function for errors, and a finite or infinite pool of items to classify, we aim at defining a service and a policy to efficiently classify items. Again for ease of expositions (especially where ML and validation are involved), we focus on tasks that require an understanding of text documents, but the concepts are generally applicable.

In this paper, we show that such a service has a few key characteristics:

1. The service must combine the crowdsourcing aspects (with the well-known challenges around it) and the efficient use of ML. As we will see there are several ways to do this, and in general, the decision on how to process each

[1]For simplicity of exposition we limit here to binary classification

item to classify depends on many factors, such as the quality of ML for that item, the expected accuracy and cost of the crowd, and the error cost structure[77]. What is common to different strategies is that the ML inference on an item, coupled with the estimated crowd accuracy in the task, may dynamically determine if we ask for crowd votes on an item, and how many (redundancy). None of this is supported by crowd platforms today.

2. At the start of a task, we neither know how well ML performs on that task, nor how well crowd workers perform. We also do not know how "fast" (how cheaply, how many labeled examples) ML can learn, something that is particularly important in a finite pool context, where there is a trade-off between the best strategy (ordering) for selecting items to label from the pool if our goal is to train a model versus if our goal is to classify as many items as possible given a fixed budget. The ability to train a model efficiently as part of hybrid classification services, be it via the recent "few shots learning" approaches or the more "traditional" active learning.

3. The key metric for the ML service component is not the accuracy but calibration [58]. Calibration has been largely neglected in ML literature compared to accuracy, but in a hybrid classification where the cost for one type of error is high (and also high with respect to the cost of asking humans), knowing if we can trust the ML service is key, just like it is important that experts tell us, whenever they make a statement, if they are sure or not. As a consequence, the ability to *estimate* what crowd and ML can do and which kind of items they can or cannot classify accurately becomes central to the effectiveness of the service.

In this chapter, we present the concepts and architecture of a hybrid classification service and then dig deeper into understanding how the ML service component, when used in a crowdsourcing context, can be trained to achieve model calibration and reduce the calibration error.

2.1 Problem Statement

We formulate the problem as follows. Given a set I of items to classify[2], a cost function λ (that includes the cost of false positives, false negatives, and cost of asking humans), an ML algorithm m and a crowd of human labelers $h \in HL$, we aim at devising a hybrid classification policy (and a supporting system architecture) that classify the items with minimal loss.

While the problem statement is general, the interesting cases - which correspond to many real-life situations - are those where the cost of machine classification is low, the cost of errors (at least one among false positive and false negative) is high, or very high, and the cost of asking humans sits in between. In other words, if C_h is the cost for a human label, C_m is the cost for a machine inference, C_{fp} is the cost for a false positive, and C_{fn} is the cost for a false negative, we expect that:

$$C_m << C_h << max(C_{fp}, C_{fn}) \tag{2.1}$$

And in fact, in most cases, it is safe to assume that C_m is very small so that for practical purposes we can consider it negligible. Assuming that to classify an item i we ask for n crowd votes on that item, then our loss function is:

$$\lambda = C_m + n \cdot C_h + \begin{cases} 0 & correct\ classification \\ C_{fp} & false\ positive \\ C_{fn} & false\ negative \end{cases} \tag{2.2}$$

Notice that this is a more general form of the problem we stated in the introduction, where we do not assume that humans are perfect oracles, which is why we may need multiple opinions. Instead, for simplicity, we assume all persons have the same costs, while in real settings different levels of competencies may command different prices.

[2]Again, for simplicity we assume binary classification but the concepts are generally applicable

Given the loss function above, if we can compute, for each item i, the probability $P(i \in pos)$ of an item belonging to the positive class, then we can estimate the expected loss $E[\lambda]$ if we classify the item as positive (and similarly do the same if we classify the item as negative). This is simply the cost independent of our classification decision ($C_m + n \cdot C_h$, which is not a random variable) plus the loss due to a possible classification error $E[\lambda_e]$ which is $C_{fp} \cdot (1 - P(i \in pos))$. For each item, we then choose the classification that minimizes such expected loss. As $P(i \in pos)$ gets close to either 0 or 1, the part of the expected loss due to classification errors gets closer to zero. In a hybrid classification service, the way we get such probability closer to the extremes is to either train "better" models or ask for more crowd votes.

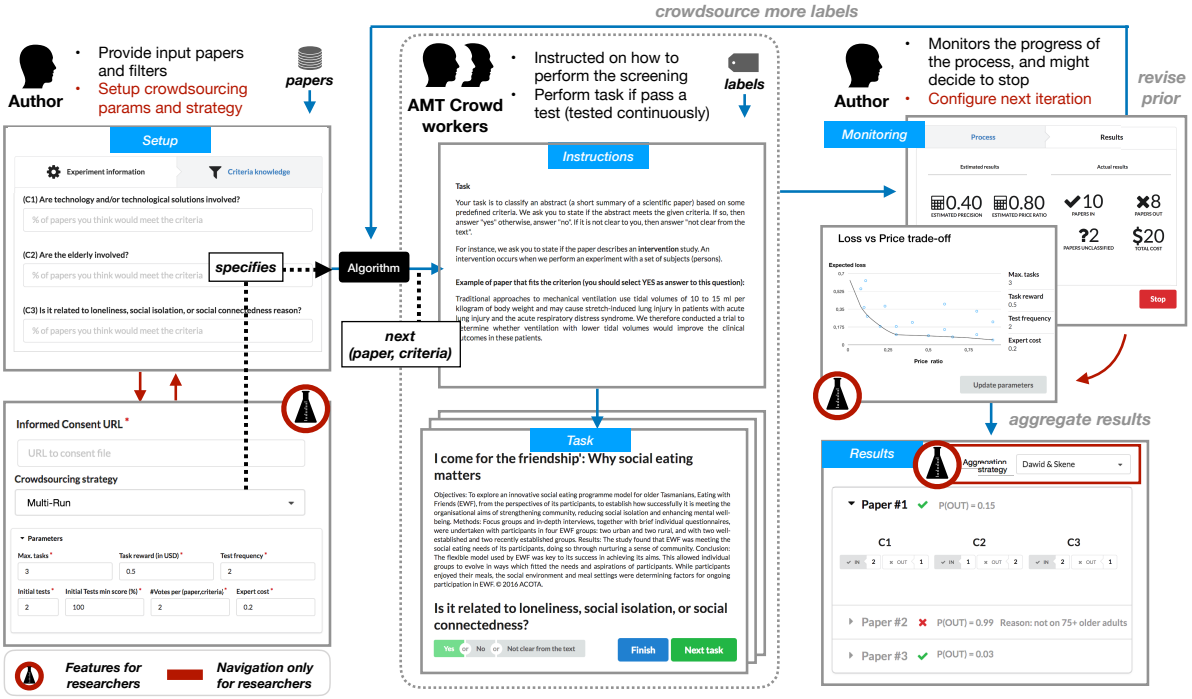


Figure 2.1: Example task workflow, modeled for a crowdsourced systematic literature review task. On the left column, the task designer writes the classification questions they want workers to answer, and point to the informed consent URL, along with other crowd task parameters. The center column shows what the worker would see in a task and the instructions given to them. The right side shows cost estimates that a platform can provide to the task designers after an initial crowdsourcing run. Details are provided in the technical report [109]

As an example, consider Figure 2.1, which shows (center of the figure) a crowd task aiming at classifying if papers satisfy a search criterion - and, in this case, if they satisfy all of a set of search criteria. Here we can assume a cost for asking a scientist or set of crowd workers to do the filtering, a cost for a false positive (including the paper in our search results), or a false negative, that is, missing the paper while it is, instead, relevant.

2.2 Approach and Service Architecture

Figure 2.2 shows how we envision a hybrid classification service. A crowdsourcing platform (CP) provides basic access to users and manages payment. There are several such platforms[3], with fairly similar features, and we do not discuss them further here. While in most cases the CP is also responsible for "serving" the specific items to users (workers) for labeling, here we need to take control and override that logic because we want hybrid classification to come into play. Specifically, we want to be able to decide for which items we ask crowd labels for, and to how many persons. We also want to dynamically stop asking if the combination of ML and the votes obtained so far gives us enough confidence to make a decision. Therefore, the items are loaded externally to the CP, into an item pool (on the right in the figure).

Processing proceeds in batch (also because we typically need to show batches of items to workers who label them in quick succession - making crowd workers wait is not a viable option). The optimal batch can be selected via an "active learning" component which is peculiar in this case because we have two sometimes conflicting optimization objectives, that of asking votes to classify items in the pool vs asking votes based on what is best for training an ML model that can then help us. This is somewhat more nuanced than traditional active learning approaches and we discuss details in the report[77].

[3]E.g., https://www.mturk.com or https://toloka.ai/

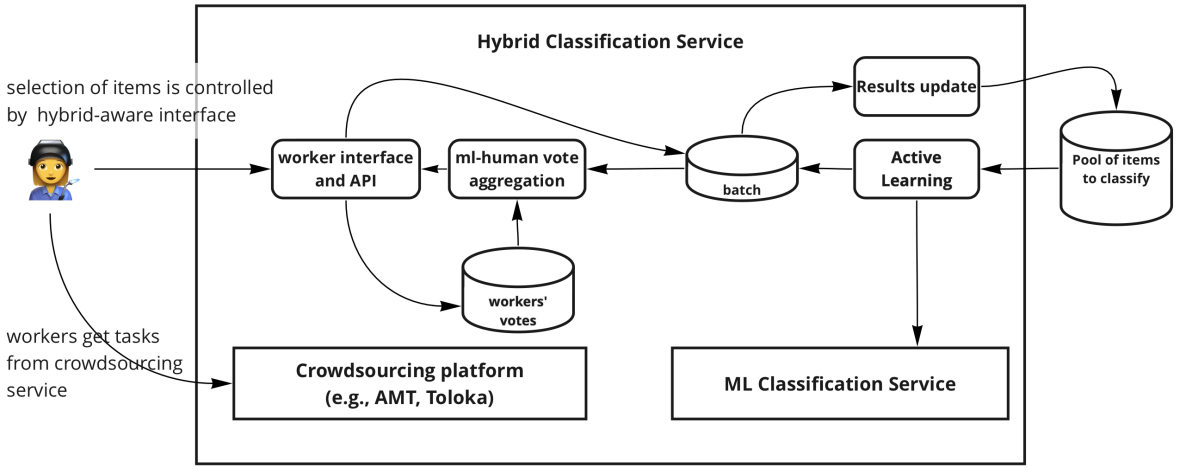


Figure 2.2: Hybrid classification service.

Once we do have a batch of votes, we then classify items, through a combination of humans and ML. As widely done in crowdsourcing, we can leverage the many existing techniques that compute workers' accuracy and aggregate votes on an item, using a variation of the Expectation-Maximization algorithm, such as Dawid-Skene [44]. However, the approach we take is to consider the classifier as a "voter", much like human workers, that is, characterized by its own "accuracy", where the accuracy (unlike workers) varies per item and corresponds to the prediction confidence on that item. Once we have the votes and accuracy we predict the class probability for each item, by first computing the probability that an item is positive (or negative) via a simple application of the Bayes rule:

$$P_{crowd}(i \in pos|V_i) = \frac{P(V_i|i \in pos) \cdot P(i \in pos)}{P(V_i)} \tag{2.3}$$

Where V_i is the set of labels given by crowd workers on item i. Given the workers' accuracy (and therefore the probability that each worker is correct given that the item is positive or negative), these quantities can be easily computed as shown for example in equation 4 of [76]. The same paper also estimates how probability changes if we ask for more crowd labels on i. A hybrid

classification service also has the benefit of machine prediction confidence (or, analogously, of having the classification service return the probability of the item being positive). We denote this with $P_{ml}(i \in pos)$. At this point, the probability of interest $P(i \in pos)$ can be computed as the average of P_{crowd} and P_{ml}[4]

Notice that in some cases the classifier may be so confident that its vote is sufficient and we do not need to resort to any crowd vote. Notice also that while workers' accuracy is independent of the items (we do not assume the crowdsourcing tasks ask for probabilities, as that is not commonly done and it would raise a whole set of challenges), ML classifiers typically do output a per-item probability and we make use of that here.

Figure 2.1 shows an example task workflow. This example shows a crowd task that performs a systematic literature review and where the strategy for selecting items to offer crowd workers is part of the task definition and controlled by the hybrid service rather than by the CP because of whether we need more labels for an item depends on whether the confidence we have thus far based on ML and current votes and the consequent expected loss. The interface also shows a component that aims at estimating the cost of classifying items in the pool.

As it is now apparent, the correct estimation of the classifier confidence for each prediction is central to minimizing classification loss. Doing so requires going into how ML algorithms can train well-calibrated classifiers in the presence of the crowdsourced label. What is interesting - and difficult - here is that in such a service we have a complex interplay between the information we obtain for the crowd (which is the only source of "truth", although imperfect - we do not assume crowds are oracles), the training of the ML model, and the calibration error. This interplay in hybrid classification services is the focus of the remainder of the chapter. We analyze and discuss this topic in the next sections.

[4]Possibly weighted, though this requires determining the appropriate weights which is part of our future work.

2.3 A Recap of Calibration

Hybrid classification is progressively gaining traction with more and more papers now starting to focus on combining human and ML inputs in classification problems. Initial work focused on very interesting ways to do this, from learning crowd vote aggregation models from "features" of the crowd task [71], to leveraging crowd to learn features of ML models, as in the brilliant paper by Bernstein and colleagues as well as others [35, 114]. In other works, automation is used to support the crowd in a variety of task-specific ways. For example, Lasecki and colleagues show an effective approach to support the crowd in speech captioning by determining speech segment lengths optimal for each worker and merging partial input of each worker [80]. These efforts are complementary to our work since we do not aim at finding features or assisting workers in performing a task. Both tasks and ML algorithms are black boxes.

More recently, proposals have emerged based on training an ML model for a task and then first using that model to classify, then asking humans if that model's confidence is not high enough. For example, Callaghan et al [29] combine ML and crowd by automatically labeling items for which the ML confidence is above a defined threshold, and by polling the crowd for the remaining ones. Variations of this approach are applied in various fields, even in fashion [5].

This belief informs their crowd classification strategy by progressively adjusting the number of votes requested on each item based on whether the crowd confirms or negates such belief. No prior assumption is made on ML classifiers' accuracy, and the hybrid algorithms are designed to be robust to weak classifiers. Finally, Nguyen et al. [100] leverage ML to identify which items to ask votes for, and whom to ask (experts or crowd). Each time their hybrid algorithm needs to pick up an item to classify and a type of human classifier (crowd or expert), it computes the *value* of each (item, classifier type) alter-

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

native by estimating the reduction in overall classification loss due to the new votes obtained divided by the cost of obtaining that loss reduction (experts are more expensive). The loss reduction is due to two reasons: items become classified by crowd or expert (both assumed to be more precise than ML, which is not used to take classification decisions once expert or crowd opinions are provided), and the additional information is used to train ML which in turn should ideally reduce the uncertainty and error on the items yet to be classified.

From this prior work, we borrow the general idea of hybrid classifiers and the specific mechanisms to use ML output in crowd classification, but our contribution lies in how to leverage the per-item prediction confidence within a service and how to support model calibration in crowdsourcing contexts.

To tackle model calibration, [131] propose to adopt *label smoothing*, where "hard" (1-0) class labels used in cross-entropy loss are smoothed into a probability distribution across classes. Such an approach has shown to be effective [58, 97, 156, 155], particularly for NLP, speech, and vision tasks [33, 36, 34, 95]. However, label smoothing and its effectiveness are still to be studied and understood, since probability amortization in output targets can bring extra noises [89] and prior art does not provide insights into how to set label smoothing hyper-parameters.

The remainder of this chapter studies the effect of label smoothing and soft targets on model calibration in NLP tasks when training labels are crowdsourced. The typical approach to deal with crowdsourced data is to aggregate votes for each item into a "gold" label, often using majority voting or expectation-maximization methods [143, 66]. An alternative approach is that of creating an empirical distribution over crowd votes. This has been referred to as the *soft target* approach [147, 11] and has shown to be effective in improving model performance in image classification [107] and emotion recognition from audio [154, 5].

While existing work has found that model performance is sensitive to noise

in soft targets [149], none has investigated the effect of label fusion methods (widely used in crowdsourcing to aggregate human answers into one label, e.g., D&S [44] and GLAD [144]) on model training.

2.4 The effect of label smoothing and soft targets on model calibration in NLP tasks

2.4.1 Background

We consider multi-class text classification task with L classes $\{1, 2, .., L\}$ where a classifier predicts $p(l|x)$, the probability that document x belongs to class l. We assume that, as commonly done, training aims at minimizing cross-entropy loss $loss = -\sum_l^L log(p(l|x)) \cdot \pi(l|x)$, where $\pi(l|x)$ here takes 1 if l is equal to the true label l^* and 0 otherwise, i. e., $\pi(l|x)$ here is the *hard target*.

Label Smoothing. In deep learning, label smoothing can be considered as a regularization technique for preventing the model from overfitting and from becoming overconfident in the classification decisions [131]. Thus, the ground-truth label distribution can be smoothed by adjusting $\pi(l|x)$ as follows:

$$\pi_{ls}(l|x) = \begin{cases} (1-\alpha) + \frac{\alpha}{L}, & l = l^*, \\ \frac{\alpha}{L}, & l \neq l^*; \end{cases} \tag{2.4}$$

where $\alpha \in [0, 1]$ is a hyper-parameter determining the amount of smoothing.

2.4.2 Our soft target approaches.

We now introduce our approaches for soft targets leveraging crowd labels. When a requester crowdsources a dataset, it is common to collect several crowd labels per sample that allow us to infer the probability distribution over classes. For each pair (x, l), we can compute probabilistic confidence of sample x has ground-truth label l using a label fusion technique $\mathcal{F}$ that aims to map crowd

votes into a class decision or a probability distribution π_f across classes, typically based on trying to estimate workers' accuracy A and factor it in while aggregating crowd votes. Formally, $\mathcal{F}$ is a function that takes crowd labels as input, and outputs a set of workers' accuracies A and a probability assignment π_f over the classes for every sample:

$$\mathcal{F}: \textit{ crowd labels} \rightarrow \langle \pi_f, A \rangle, \tag{2.5}$$

Common examples of fusion methods are, e.,g., GLAD [144], or D&S [44]. By doing so, we can create *soft targets* according to Eq. 2.5 and incorporate them in the loss function. Note that now $\pi_f(l|x)$ comes from the 'natural' distribution of crowd-contributed labels as well as fusion methods, rather than from arbitrary smoothing with pre-selected hyper-parameters as in label smoothing methods. Furthermore, *each data point is smoothed differently, according to the ambiguity as perceived by the crowd.*

We further propose *semi-Hard* (*sHard*) method that does the one-side smoothing, that of the most likely label as identified by the fusion method. This can be seen as a soft target approach but also as a hard one where samples are weighted by the ground truth probability of the most likely label (note that in this case $\pi(l|x)$ is no longer a valid distribution):

$$\pi_{sh}(l|x) = \begin{cases} \pi_f(l|\textit{votes}, F), & l = l^*, \\ 0, & l \neq l^*, \end{cases} \tag{2.6}$$

Figure 2.3 shows an overview and comparison of hard targets, label smoothing, soft targets, and semi-Hard targets.

2.4.3 Effect of label fusion techniques

Figure 2.4 shows an example case on the Reuters-21578 dataset to highlight the effect of using different label fusion techniques for label aggregation.

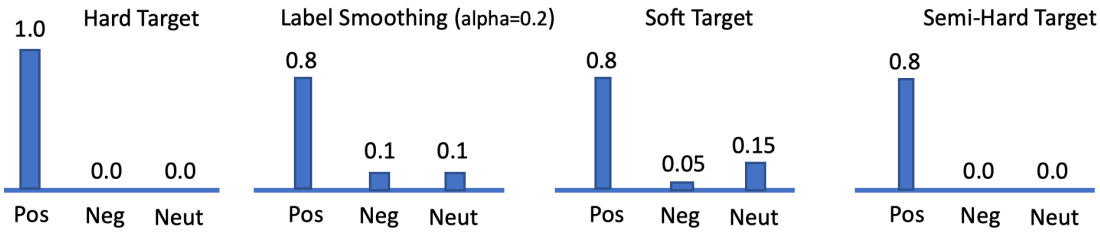


Figure 2.3: Different target mass functions for classification problem with three classes (Pos- positive, Neg- Negative, Net- Neutral classes). The Hard Target is equivalent to one-hot label encoding, the distribution for Soft Target is obtained from crowd votes via a label fusion method (e.g., MV or DS).

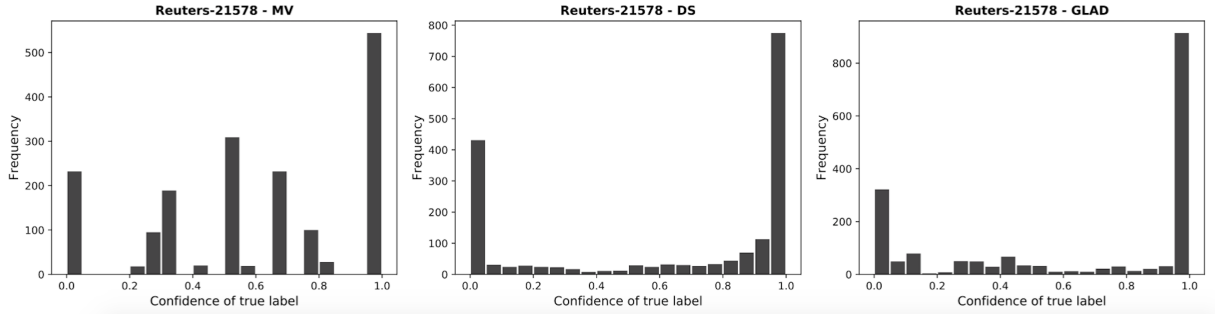


Figure 2.4: Probabilities assigned to correct classes (Ground Truth) by MV, DS, and GLAD label fusion methods on Reuters-21578 dataset.

2.4.4 Contribution

The *crowdsourcing-label fusion-training* pipeline is commonly used and addressing the impact on calibration is, therefore, crucial to be able to generate "trustworthy" models. Specifically, in this work, we propose soft target methods that can incorporate any label fusion method: we present label fusion methods that accept raw crowd votes and output label probability distributions for every sample in the dataset, and where these label distributions are then used as soft targets to train well-calibrated neural models.

We evaluate our approach on 13 crowdsourced datasets and evaluate the effect of both soft targets and label smoothing in training the multi-layer perceptron and DistilBERT, a deep transformer model [119]. Our results show that soft targets are more effective for model calibration than label smoothing. We

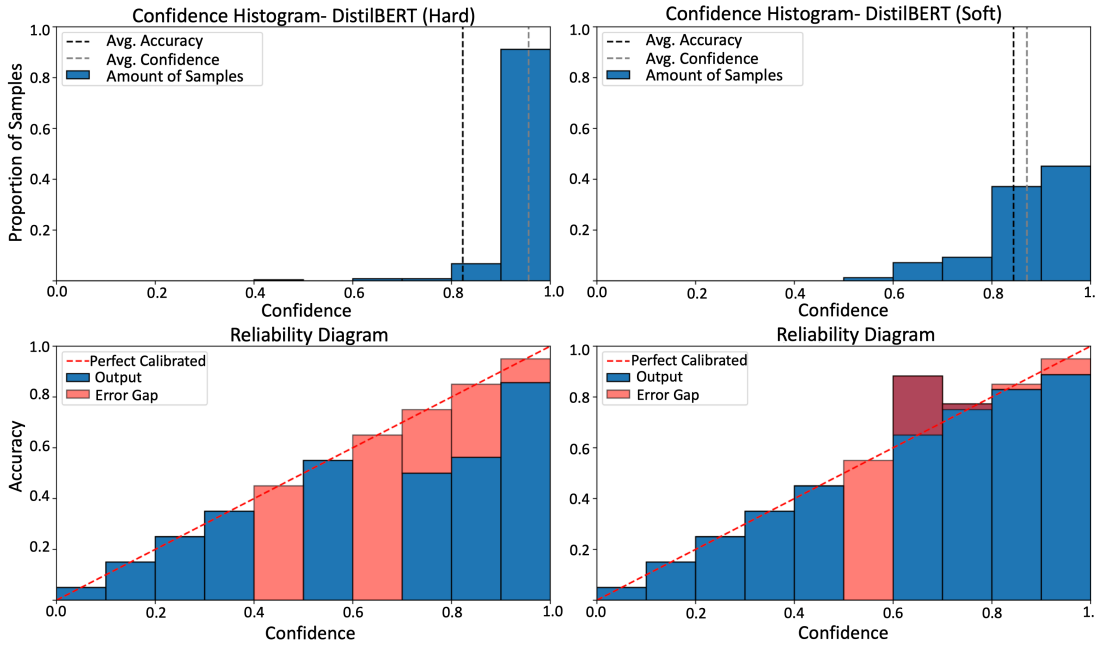


Figure 2.5: The effect of soft targets on probability calibration for DistilBERT on Deaths-in-India dataset.

further demonstrate that our proposed soft target methods substantially improve both model performance and probability calibration across datasets of different noise levels, and this improvement is more obvious when they are used to train the deep transformer model. An example is visualized in Figure 2.5, which shows the distribution of prediction confidence from DistilBERT trained with hard labels (left figures) and soft targets (right figures): the average prediction confidence better matches the average accuracy in the case of soft targets (top right) and the confidence closely approximates accuracy across different confidence levels (bottom right). Results support the fact that deep models tend to become over-confident and not calibrated [58] when they are trained with hard targets; top left figure shows that the average accuracy of the model is significantly less than the average confidence. This results in having larger error gaps in the reliability diagram (bottom left).

2.5 Experimental Setup

2.5.1 Datasets

Our study relies on 5 binary and 8 multi-class crowdsourced datasets. As finding public crowdsourced datasets with individual crowd votes is challenging, we chose ten datasets (Table 2.1) provided by Appen[6] for different text classification tasks. These datasets come from text-based crowdsourcing tasks (mainly classification), but the only "ground truth" information available comes from aggregating workers' votes. Neither the individual votes nor the details on how the crowd experiment was run are available.

Dataset	Class	Train/val/test	Noise ratio
1. First GOP debate sentiment analysis (GOP2-sentiment)	2	6195/ 576/ 628	0.07
2. Disasters on social media (Disaster-relevance)	2	7557/ 740/ 684	0.09
3. Do these chemicals contribute to a disease? (Chemicals&Disease)	2	3088/ 186/ 162	0.16
4. Economic news article tone and relevance (News-relevance)	2	5098/ 452/ 526	0.13
5. Corporate messaging (Corporate-messaging)	3	2615/ 120/ 117	0.08
6. First GOP debate sentiment analysis-Sentiment (GOP3-sentiment)	3	11669/ 540/ 348	0.19
7. Twitter sentiment analysis: Self-driving cars (Self-driving-cars)	3	5399/ 150/ 147	0.04
8. Drug relation database (Drug-relation)	3	1866/ 75/ 72	0.01
9. Indian terrorism deaths database (Deaths-in-India)	3	25622/ 300/ 237	0.20
10. GOP tweets subject categorization (debate-subject)	5	1206/ 150/ 135	0.04

Table 2.1: Dataset properties.

This situation required us to examine the ground truth we had and assess how noisy are the aggregated labels. Two researchers manually re-annotated each dataset to analyze how accurate the labels are. We checked the content (eg., tweets) and annotated it. We observed that on average, we do not agree with 10% of the aggregated crowd labels (and up to 20% for some datasets!). This indicates that crowdsourced labels can be noisy even after collecting multiple votes per sample. Our manual annotations of test datasets are presented in the (anonymized) project repository[7].

[6]https://appen.com/resources/datasets/

[7]https://github.com/Evgeneus/Label-Smoothing-in-Text-Classification

The aforementioned datasets give us only one statistic (i.e., confidence) about the distribution of labels[8]; however, we wanted to explore if having individual crowd votes rather than the aggregated ones and employing different label fusion algorithms might give us more insights. To this end, we use three additional textual datasets with actual crowd labels as well as the ground truths: i) Movie-Reviews (3-classes) [113], ii) Reuters-21578 (8-classes) [112], iii) Amazon-Reviews (binary)[9].

2.5.2 Configurations and Training

We evaluated: i) a simple one-layer neural network (NN1) with text vectorized via tf-idf, and ii) fine-tuned the DistilBERT model (D-BERT) [119] (6 layers, 768 hidden dim, 12 heads, 65M parameters) to compare the behavior on different models. We trained them with the cross-entropy loss using: i) *hard* targets (one-hot encoded labels), ii) *label smoothing* (Eq. 2.4), and iii) the proposed *soft* (Eq. 2.5) targets. We tested the impact of three label fusion methods: i) Majority Voting (MV), ii) D&S [44], which models worker reliability, and iii) GLAD [144], which further considers the task difficulty. Following [97, 138], we considered $\alpha = 0.05$ and $\alpha = 0.1$ for label smoothing.

For each (NN1, training target) configuration, we performed a grid search for the following hyperparameters on the validation set: minimum document Frequency, number of TF-IDF features, word n-gram range, learning rate, weight decay, and class weights. For each (D-BERT, training target) configuration, using the validation set we searched for learning rate and a few options for class weights depending on the class distribution; the rest of the parameters remained as default. The networks were trained using the ADAM optimizer iterated up to 500 epochs for NN1 and 100 epochs for D-BERT with early stopping. We set the batch size of 32 for D-BERT and the sequence length of 512 WordPiece

[8] https://tinyurl.com/Calculate-a-Confidence-Score

[9] https://github.com/TrentoCrowdAI/crowdsourced-datasets

tokens. The experiment details, datasets, and source code are available online[7].

Evaluation Metrics: To evaluate both prediction and confidence quality for the classification models, we use two metrics: i) macro F1 score and ii) ECE, which measures the difference in expected accuracy and expected confidence [98]. The smaller the ECE score, the better the calibration of the model.

2.6 Results

2.6.1 Label smoothing vs soft targets

Table 2.2 reports the results of label smoothing and soft targets with NN1 and D-BERT. Across the ten Appen datasets, results are inconclusive: soft targets provided comparable results in terms of F1 while gave mixed results for ECE across the datasets: in many cases label smoothing collapsed ECE score in the NN1 model (by 0.9% for α=0.05 and 1.5% for α=0.1 on average). The same applies independently of how we encode text (eg, the cited additional material report on LSTM encoding)[7].

	1.GOP2-sentiment		2.Disaster		3.Chemicals&Disease		4.News		5.Corporate Messaging		6.GOP3-sentiment		7.Self-driving cars		8.Drug Relation		9.Deaths in India		10.Debate-subject	
	F1	ECE	F1	ECE	F1	ECE	F1	ECE	F1	ECE	F1	ECE	F1	ECE	F1	ECE	F1	ECE	F1	ECE
NN1-Hard labels	78,5	**11,0**	82,4	**9,8**	74,5	20,4	75,1	13,9	89,8	**3,2**	65,3	6,4	66,7	6,3	66,0	7,3	**83,2**	7,4	88,2	4,0
NN1-Soft	**79,7**	18,3	**82,9**	20,2	70,4	**6,4**	**75,8**	15,3	86,4	5,4	**66,6**	**4,9**	66,8	5,8	71,1	3,6	82,1	5,8	**88,9**	**2,1**
NN1 ($\alpha = 0.05$)	73,4	14,3	82,5	17,6	**76,5**	18,6	73,4	14,3	**90,6**	5,2	64,9	7,5	70,4	**5,0**	**73,7**	**3,0**	80,5	**4,8**	88,1	7,4
NN1 ($\alpha = 0.1$)	73,7	13,2	82,3	19,0	68,7	13,6	73,7	**13,2**	**90,6**	9,8	65,5	8,6	**70,8**	7,8	69,8	8,5	82,3	**4,8**	88,1	6,0
D-BERT-Hard labels	**78,2**	8,4	78,5	13,6	**75,2**	22,3	72,0	21,4	**89,7**	9,0	59,3	31,8	**59,0**	25,5	**80,7**	13,7	81,4	13,3	80,7	18,1
D-BERT-Soft	**78,2**	11,2	**82,3**	10,7	71,4	**16,4**	**72,7**	18,9	87,9	4,1	**63,1**	**11,1**	55,0	**17,7**	79,4	**4,9**	**84,0**	6,0	**83,7**	**3,8**
D-BERT ($\alpha = 0.05$)	59,2	18,2	79,9	12,8	65,6	25,8	72,6	**17,5**	87,1	6,7	59,9	25,3	52,3	25,2	**80,7**	8,9	81,2	10,3	80,0	15,4
D-BERT ($\alpha = 0.1$)	76,6	**3,3**	81,0	**3,5**	73,3	26,3	70,3	24,6	**89,7**	**2,3**	63,3	18,2	56,0	21,3	79,1	8,0	81,6	**5,1**	80,7	12,5

Table 2.2: Macro F1 and ECE results in % for NN1 and D-BERT with smoothed and soft targets.

D-BERT is a more interesting case as we know that calibration issues arise mainly with deep nets [58]. Although our objective is to investigate how label smoothing and soft targets perform in terms of improving the ECE score (without sacrificing F1), we observed that using soft targets can also improve the F1 score. For example, results on "Deaths in India" dataset shows that training the D-BERT model with hard targets obtains 13.3% in ECE and 81.4% in F1 while

the calibration error is reduced to 6.0% and F1 score is increased to 84.0% when we use soft targets. Overall, training D-BERT model with soft labels from crowd gives an improvement in ECE for nine datasets (7.2% of average improvement across 10 datasets), and a boost in F1 for 6 out of 10 datasets. In contrast, label smoothing also can improve ECE (from 1% to 9% depending on the chosen α parameter) but always harm F1 score.

On datasets with individual crowd labels, we can also experiment with different fusion methods. We show the results in Table 2.3. Label smoothing here shows mixed results (improves on ECE on Movie-Reviews data but does significantly worse in ECE on Amazon-Reviews where the training labels are very accurate - and we do not have results for D-BERT on Reuters-21578 as the data did not contain raw texts) In contrast, the proposed soft target method improved ECE (up to 15.7%) across all datasets. Notably, for the GLAD fusion method, the soft target method enhanced *both* F1 (up to 15.7%) and ECE (up to 7.1%) on the D-BERT model across the datasets with both high and low levels of label noise. This is in part due to the better performance of GLAD in truth inference and because GLAD generates less skewed label distributions allowing the soft method to handle the noises.

2.6.2 sHard Targets

We tested *sHard* (Eq. 2.6) targets following the configurations explained in Section 2.5.2. Results show that NN1 with *sHard* targets *always* led to improvement of ECE (1.7% on average), while F1 remained comparable to the models trained on hard labels (Table 2.4). When we compare the ECE performance of *sHard* to *Soft* and *label smoothing*, we see that *sHard* outperforms *label smoothing* on 7 datasets and *Soft* targets on 6 datasets. Training D-BERT model with sHard targets provides an improvement of ECE for 8 datasets out of ten (1.8% of average improvement across 10 datasets), and improves F1 for 7 datasets as to using *Hard* labels during the training. Finally, *sHard* improves ECE on

Model	Movie-Reviews		Reuters-21578		Amazon-Reviews	
	ECE	F1	ECE	F1	ECE	F1
NN1-Hard labels (MV)	4,2	58,5	**4,7**	63,6	**11,0**	**95,8**
NN1-Soft (MV)	**3,1**	**59,8**	5,3	65,1	12,6	**95,8**
NN1 (α=0.05, MV)	8,2	53,5	6,1	64,6	17,3	95,7
NN1 (α=0.1, MV)	10,4	55,3	5,4	**65,5**	20,8	**95,8**
NN1-Hard labels (DS)	14,3	57,6	3,6	**72,4**	11,0	**95,8**
NN1-Soft (DS)	**8,7**	**58,5**	4,5	70,9	**10,6**	95,7
NN1 (α=0.05, DS)	9,5	56,1	**2,9**	70,8	17,3	**95,8**
NN1 (α=0.1, DS)	**8,7**	55,2	4,7	69,7	20,8	**95,8**
NN1-Hard labels (GLAD)	**4,5**	**57,7**	**3,1**	67,1	**10,5**	**96,0**
NN1-Soft (GLAD)	6,6	56,9	4,1	**70,6**	10,9	95,7
NN1 (α=0.05, GLAD)	9,3	52,5	3,7	68,6	17,1	95,8
NN1 (α=0.1, GLAD)	7,0	55,7	4,4	69,6	20,6	95,8
D-BERT-Hard labels (MV)	36,5	56,3	-	-	5,9	**93,0**
D-BERT-Soft (MV)	**21,1**	54,9	-	-	**5,3**	92,4
D-BERT (α=0.05, MV)	25,7	**57**	-	-	7,3	92,4
D-BERT (α=0.1, MV)	21,6	56,7	-	-	11,5	**93,0**
D-BERT-Hard labels (DS)	34,2	56,5	-	-	3,3	92,7
D-BERT-Soft (DS)	34,1	53,6	-	-	**2,0**	**93,1**
D-BERT (α=0.05, DS)	**23,0**	**58,5**	-	-	7,7	92,8
D-BERT (α=0.1, DS)	31,6	49,4	-	-	10,3	**93,1**
D-BERT-Hard labels (GLAD)	36,1	56,6	-	-	9,2	91,1
D-BERT-Soft (GLAD)	**20,4**	**58,2**	-	-	**2,1**	**94,1**
D-BERT (α=0.05, GLAD)	29,5	54,9	-	-	7,7	92,8
D-BERT (α=0.1, GLAD)	24,5	55,6	-	-	12,8	93,4

Table 2.3: Performance of NN1 and D-BERT with targets obtained from different fusion methods.

	1.GOP2-sentiment		2.Disaster		3.Chemicals&Disease		4.News		5.Corporate Messaging		6.GOP3-sentiment		7.Self-driving cars		8.Drug Relation		9.Deaths in India		10.Debate-subject	
	F1	ECE	F1	ECE	F1	ECE	F1	ECE	F1	ECE	F1	ECE	F1	ECE	F1	ECE	F1	ECE	F1	ECE
NN1-sHard	**79,8**	**6,8**	**82,9**	**9,2**	71,3	**15,4**	74,9	**11,1**	**90,6**	**2,0**	**66,0**	**5,8**	66,0	**4,2**	66,0	**5,6**	82,9	**4,6**	**88,9**	**3,4**
D-BERT-sHard	76,8	**5,5**	**80,5**	**12,4**	**77,1**	**20,3**	**72,6**	**15,9**	**90,6**	**7,6**	**60,1**	**28,1**	57,0	27,0	80,7	**11,9**	**86,2**	**9,3**	78,7	20,7

Table 2.4: Macro F1 and ECE results in % with sHard targets; bold numbers shows sHard outperforms hard targets.

Comparison vs Hard Target	ECE (better of equal)	F1 (better of equal)
D-BERT + sHard	8 out of 10 datasets	7 out of 10 datasets
D-BERT + Soft	9 out of 10 datasets	6 out of 10 datasets
D-BERT + Label Smoothing (alpha=0.05)	8 out of 10 datasets	4 out of 10 datasets
D-BERT + Label Smoothing (alpha=0.1)	8 out of 10 datasets	5 out of 10 datasets
D-BERT + Label Smoothing (alpha=0.15)	10 out of 10 datasets	5 out of 10 datasets
D-BERT + Label Smoothing (alpha=0.2)	10 out of 10 datasets	5 out of 10 datasets

Comparison vs Hard Target	ECE (better of equal)	F1 (better of equal)
NN + sHard	10 out of 10 datasets	6 out of 10 datasets
NN + Soft	6 out of 10 datasets	7 out of 10 datasets
NN + Label Smoothing (alpha=0.05)	4 out of 10 datasets	5 out of 10 datasets
NN + Label Smoothing (alpha=0.1)	3 out of 10 datasets	4 out of 10 datasets
NN + Label Smoothing (alpha=0.15)	2 out of 10 datasets	5 out of 10 datasets
NN + Label Smoothing (alpha=0.2)	2 out of 10 datasets	4 out of 10 datasets

Figure 2.6: The comparison of different target methods against models trained on Hard Targets for 10 datasets from Appen.

2 datasets that have individual crowd votes while remaining comparable on 1 dataset (Table 2.5).

Model	Movie-Reviews		Reuters-21578		Amazon-Reviews	
	ECE	F1	ECE	F1	ECE	F1
NN1-sHard (MV)	4,3	**60,2**	5,1	**64,3**	**10,3**	95,5
NN1-sHard (DS)	**10,5**	**58,2**	3,7	72,3	**10,4**	95,7
NN1-sHard (GLAD)	5,7	**58,6**	3,4	67,1	10,8	95,8
D-BERT-sHard (MV)	**35,9**	**57,4**	-	-	**4,9**	92,5
D-BERT-sHard (DS)	**30,0**	**57,9**	-	-	3,3	92,1
D-BERT-sHard (GLAD)	**35,2**	56,4	-	-	**5,1**	**93,3**

Table 2.5: Performance of NN1 and D-BERT with sHard targets obtained from different fusion methods.

2.6.3 Key findings

Figure 2.6 and Table 2.6 summarizes the results by showing the average improvement of the models trained on sHard/Soft/Smoothed targets to the models trained on hard targets (our baselines). Results show that our proposed *Soft* and *sHard* target methods substantially improves the model calibration.

2.7 Limitations - and the Road Ahead

This work scratches the surfaces of hybrid classification services and the science behind them. We have shown the centrality of calibration in contexts where the

Model	F1	ECE	Model	F1	ECE
NN1-sHard	0.04	**-1.7**	D-BERT-sHard	**0.6**	-1.8
NN1-Soft	0.1	-0.7	D-BERT-Soft	0.3	**-7.2**
NN1 (α=0.05)	**0.3**	0.9	D-BERT (α=0.05)	-3.6	-1.1
NN1 (α=0.1)	-0.4	1.5	D-BERT (α=0.1)	-0.3	-5.2

Table 2.6: Avg. improvement in % to hard labels.

loss function is skewed and where the cost of errors is high compared to the cost of asking humans. We have also shown the effect of soft and semi-Hard targets in text classification with crowdsourced data, across several datasets and fusion methods. The effect on calibration error and the benefits of the proposed approach are manifest for deep models, that are known to be more affected by calibration issues. While promising, the initial work requires deeper investigations: we need to expand the experiment to other deep network architectures and get a deeper understanding of what drives the behaviors we are seeing. The same is true for label fusion methods and related datasets and classification problems, especially the classification problems with a high number of classes and varying degrees of noise. Finally, we need to integrate the modules into a hybrid classification service and test it "end to end", progressively building science that can eventually put classification tasks on autopilot so that crowd and ML, and their integration, become a commodity rather than an art and a hard challenge.

Chapter 3

The science of rejection

For decades, the primary way to develop and assess machine learning (ML) models (and one of the primary avenues to publication) has been based on *accuracy* metrics (e.g., precision, recall, F1, AUC). The need to beat leader-boards when publishing, with public datasets and rankings, to some extent exacerbated this focus on accuracy metrics, at least for classification models. There is nothing bad in efforts to improve model accuracy: they should stay among the main goals of ML research. But we argue that one of the reasons for the disconnect between the amazing progress of ML research (and the corresponding expectations of professionals in any field that are now sky high) and the limited adoption of ML in the enterprise is the focus on one aspect of the problem only, and on the fact that we have not paid enough attention to how and why models are used in practice, and to the aspects and metrics that are relevant to enterprises when they adopt and deploy a model.

In this chapter, we take an intentionally provocative stance and state that *accuracy metrics are optional*, desirable properties of an ML model that are sometimes marginal with respect to other metrics rarely addressed in the literature, if at all. To this end, we start by taking a critical look at the use of ML models in typical enterprise scenarios, and from there abstract a simplified but general AI workflow followed in practice. We then present an analysis of how

current metrics— both the accuracy metrics and some more recent proposals— are misaligned with the value and cost induced by the workflow. *We come to the conclusion that what we need is a new set of cost (loss) metrics as well as a science for learning when to reject the inferences done by an ML model - and correspondingly for identifying subsets of items where we can trust the model.*

We then switch our discussion to the scientific progress related to those problems: where are we now and what can human computation do? We start by reviewing the ongoing work in ML and human computation, including the recent effort on data excellence and hybrid human-machine systems. We argue that human computation can play a lead role in providing methods for model rejection, yet discussions on this topic are largely missing. We discuss opportunities for research on metric definition, ML failure detection and characterization, and building the rejector, all involving crowds.

3.1 AI Workflows and the Metrics that Matter

Figure 3.1 shows the typical way in which AI solutions are deployed in an end-to-end enterprise workflow. First, we either reuse or fine-tune a pre-trained model or use it in combination with some task-specific model. Either way, we have some ML classifier m that, given an input $i \in I$ (where I is a possibly infinite set of items to classify), produces a predicted class and a confidence (or a distribution of predicted class with confidences). Almost invariably today in any model deployment there is then the filtering based on whether the confidence is greater than some threshold, and if so the prediction is applied, else a default path is followed - most often the very same path that was there before the introduction of AI in the process. This is true from reading Xrays to detecting intents in a chatbot to automatically routing requests and on to nearly any use case we can imagine.

From this simple description we can draw a few observations: First, the

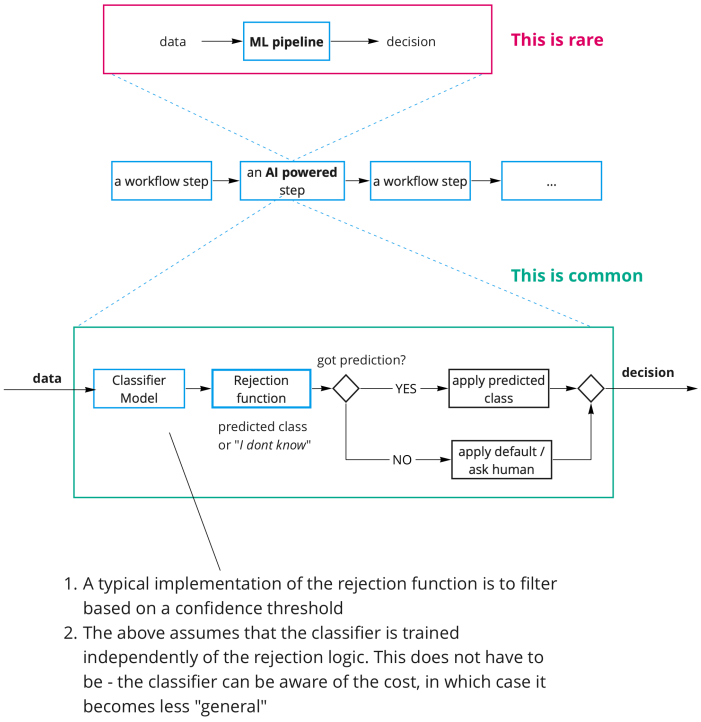


Figure 3.1: Typical implementation of ML models into an enterprise workflow.

threshold and the system behavior depend on the "cost" of machine errors and its relation to the cost of a rejection and the value of a correct machine prediction. Let's refer to the value of a correct prediction as V, to the value (cost) of following the default flow as C_d and to the cost of a wrong prediction as $C_w = K \cdot C_d$ (that is, we express C_w in terms of how "bad" is an erroneous prediction compared to the default flow). Also, for simplicity, let's assume that $V = -C_d$ (this is both reasonable and to simplify the presentation, but none of the concepts below depends on this assumption), and let's normalize by taking $V = 1$, again for simplicity. If the enterprise has a sense for the value of K, then the confidence threshold descends from that, and so does the expected utility we get from each execution of the workflow. The actual math and values are not relevant to the discussion, but simple math (omitted) shows that the optimal threshold T is $T = \frac{K-1}{K+1}$, *assuming the model is well calibrated*. Similarly, we can show that the *expected value* for each prediction with confidence c is

$$E[value] = -1 \cdot \rho_t + (1 - \rho_t) \cdot (c(K+1) - K) \tag{3.1}$$

where ρ_t is the probability of a prediction confidence being below the selected threshold t.

The second observation is that if we have a well-calibrated model with *arbitrarily bad accuracy* α, we can still get value from it. Having no model at all gives us a value of $-n$ to process n items (remember $C_d = -1$). Having a really inaccurate (but well-calibrated!) model that, in the rare proportion of cases hc where it has high confidence is correct, gives us a value of $-n \cdot (1-hc) + n * hc \cdot$, which is higher. In theory hc *can be arbitrarily small and we still obtain value*. In practice this is not the case as the decision to adopt AI comes with development, deployment, testing and management costs so there is some minimal value below which AI adoption does not make sense.

Notice that commonly adopted measures of calibration errors, such as the Expected Calibration Error (ECE) and its variation [101] (eg, based on how we bin the samples) are not part of the above formulas and it is easy to show that they do not correspond to the metric we want to improve.

In this, it is important to point out that we are not just referring to far-fetched corner cases. The "problem" with commonly adopted metrics of calibration error is that, while they are valuable in helping us to get a sense of the model calibration as a whole and they are independent of any threshold T or cost structure, that is also their limitation. Indeed, when we apply a model as per the workflow above we only really care about calibration around the confidence threshold T, which either would make predictions we (incorrectly) reject above the threshold or that make predictions we hazardously accept below the threshold, where they belong. It is not uncommon for calibration techniques such as temperature scaling to show spectacular ECE results but if our threshold is 0.8, we don't care about the error in the 0.1-0.2 range, nor do we care if the confidence is 0.999 or 0.85.

Other measures of calibration errors seem more meaningful in this regard: for example, we can measure the difference between *expected value* as per the formula above (which does not depend on prediction correctness, only on confidence values) and the *actual value* measured on a test set given the threshold T based on actual correctness of predictions. If the model confidences represent probabilities and the model is calibrated, then the difference is only due to statistical sampling error, and as the test dataset increases in size, it should go towards zero. If the model is not calibrated, then the difference has two sources: the error due to calibration and the error because T was set based on optimal calibration. Another approach is to select the threshold T_v not based on the above formula but empirically based on a validation set, by picking the threshold that minimizes the cost over that dataset. In this case, the difference between the theoretical threshold T (which assumes calibration) and T_v grows as the calibration error grows. These are just two examples of metrics around which we have not developed a theory yet but that seems promising in terms of defining a notion of calibration error that, when minimized, leads to a higher expected value.

Last but not least, the same discussion we had above for accuracy (that is: it's ok if we are accurate for an arbitrarily small subset of items) also applies to calibration! In principle, if we had a magic way to know that our model m is well calibrated for a subset $I_c \subseteq I$ of items, and we knew, given an item i, how to tell if $i \in I$, then we would have a useful model, no matter what our cost structure is, no matter the accuracy, and no matter the overall calibration over I.

At some point the problem becomes recursive and we need to draw the line (knowing I_c is kind of the same thing as saying that we are confident about our confidence measures). But the main point remains: the better we can identify the subset of items for which our model m is calibrated - according to the metrics defined above - the lower is the cost for our deployment of m in an AI workflow.

Now, because in this chapter we assume that m is given to us and that the decision to accept or reject examples is done downstream - as it often is in reality, this means that we need to equip AI practitioners with a "*science of rejection*" that helps build acceptance or rejection logic for each prediction - either by recalibrating a model and/or by identifying areas of I where we can and cannot trust m.

3.2 Where Are We Now?

Confidence Calibration aims at making the model prediction confidence to be representative of the likelihood of the prediction being correct. Typical methods smooth the training labels by converting a single hard label into a probability distribution using certain heuristics [131], e.g., by reducing the probability of the label and amortizing the reduced probability over other labels. Such an approach has shown to be effective but again our concern is with the metric that the approaches optimize. Furthermore, it has recently been shown that while label smoothing can prevent neural nets from becoming over-confident, it results in loss of information about resemblances between data instances [97]. From the perspective of ML failures, label smoothing only deals with biases in the label and cannot deal with those in the feature space.

Adversarial Training instead, can reduce such biases in the feature space by generating adversarial instances [132], also called out-of-distribution instances as they are not captured in the training data. The idea is developed driven by the observation that imperceptible differences in the processed data can lead to prediction failures. The approach however can lead to a skewed distribution of the generated instances that are similar to existing training instances. In particular, for certain features that are missing in the training data, it's unlikely that adversarial training can generate such data items.

Data Excellence is a recent effort to enhance the quality of training data through

the human discovery of items that are challenging to ML models, especially the unknown unknowns (i.e., items on which the model makes high confidence errors) [7]. Unlike machines that fully rely on knowledge explicitly encoded in predefined training data, humans excel at leveraging broad, tacit, and contextual knowledge in decision-making and justification. Human computation has, therefore, emerged as a new, promising approach to detecting unknown unknowns. A seminal work by Attenberg *et al.* proposes to ask humans to gather publicly accessible instances that are potentially difficult for the model to handle [7]. Lakkaraju *et al.* introduce a data partitioning technique that first organizes the data into multiple partitions based on feature similarity, and then uses an explore-exploit strategy to search for unknown unknown instances across these partitions [79]. An important finding in human computation studies reveals that unknown unknowns often come with internal consistency, making them particularly suitable to be described by human language building on top of concepts [85].

There is recently a surge of interest in this topic from both academia and industry. HCOMP recently launched the CATS4ML challenge[1] to leverage crowdsourcing for unknown unknowns discovery; Facebook recently introduced the Dynabench platform[2] for a similar purpose.

Despite that, the recent effort has focused on data only, taking a bottom-up approach, that is: by collecting better data we hope the machines will learn what is needed. The assumption however comes without any theoretical guarantee or strong empirical evidence.

Hybrid Human-Machine Systems tackle the process of solving classification problems by leveraging both humans and machines [108, 145]. Initial work focused on very interesting ways to do this, from learning crowd vote aggregation models from "features" of the crowd task [71], to leveraging crowds to learn

[1] `https://cats4ml.humancomputation.com`
[2] `https://dynabench.org`

features of ML models, as in the brilliant paper by Bernstein and colleagues as well as others [35, 114]. More recently, proposals have emerged based on training an ML model for a task and then first using that model to classify, then asking humans if that model's confidence is not high enough [29]. The effectiveness of such an approach is, consequently, heavily dependent on the reliability of machine confidence, which has shown to be very poor, especially for deep learning [58, 17].

3.3 What Can Human Computation Do?

Our literature analysis points to the fact that research from existing efforts only provides partial solutions to desirable ML systems. The problem of model rejection has been seldom discussed in the human computation community.

The problem is closely related to the ML reliability issue that is heatedly discussed across many other communities, together with other issues such as transparency and fairness. Within Computer Science, discussions have been revolving around the relation between systems and people, e.g., the importance of human centrality. A visible trend is the fast-growing work of human-AI interaction [4, 84]. Much of those work takes the angle of humans as users or stakeholders; in comparison, the computational roles of humans in the process of better making ML systems or the functioning of hybrid human-AI systems are seemingly less discussed. We note that human involvement in the system (creation) is key to bridging the gap between the need of stakeholders and the engineering of the system, hence of great scientific relevance to the engineering communities on ML, data, and systems.

Human computation started with the very idea of leveraging human intelligence to solve tasks that are beyond the capability of automated systems, considering specifically the computational roles of humans without ignoring the personal and social properties. Responding to the model rejection problem, the

key research question is the following:

RQ: *How can human computation provide an approach to tell when machine learning systems fail?* Answers to this question can provide guidance for collecting high-utility data for model training and allow for safe decision delegation to machines. Having such an approach can, therefore, largely benefit data creation and hybrid decision-making, and together, promise a human-in-the-loop ML system that can be relied upon. We extrapolate a non-exhaustive list of sub-questions as follows:

SRQ 1: *What are the proper metrics for the cost of using ML with rejection?* Metrics should be re-considered to measure the cost-effectiveness of a hybrid human-AI system with a selective classifier whose prediction can be rejected. The following cost needs to be taken into account in the functioning of the system: 1) cost of wrong predictions by the ML model. Such cost is task-specific: the false positive and false negative should be weighted according to the task; 2) cost of human-made decisions. In the creation of the system, the cost induced by human involvement in creating the classifier and the rejector should also be considered.

SRQ 2: *How to involve the right stakeholders to report on machine learning failures?* We can imagine that not all failures are easily detectable by random crowds, especially in social contexts where the perception of the quality of the services is dependent on personal preferences or cultural background. Opening a channel where stakeholders can effectively report on their experiences is the first key step to the rejection problem. The "how" in this question is, therefore, relevant to both the "who" and "through which means".

SRQ 3: *How to effectively characterize machine learning failures?* Characterization of failures can either be done on a per-item basis, i.e., using examples as descriptions or on the conceptual level. The latter would be preferred to provide a cognizable description of "when the model fails" to developers and stakeholders.

The feasibility of such a "symbolic" approach is less a concern given the internal consistency of machine unknowns. In addition to the question of "which form" the description should be, it is also important to consider "what materials" to use in human characterization. We argue that the important factor is the involvement of models, through e.g., explainable AI techniques, such that the internal mechanism of the model can be exposed to allow for more effective identification of the failure reasons. Recent work has shown that human computation can be a favorable approach to explanation itself [16].

SRQ 4: *How to build the rejector?* A smart rejector can be built based on human feedback on machine failures. This can be done in a data-driven way like the usual ML models are trained, or through a hybrid data- and knowledge-driven method that allows for more explicit control over the items on which the prediction should be rejected. Reliability of human feedback should be considered, as to how human-labeled data has been used for ML training.

In summary, we propose a new frame for evaluation where we argue that i) rejection - and related relevant metrics - should be a first class citizen of ML research, both theory and practice, that ii) hcomp is a promising way to go, but that iii) it requires very different methods than human computation for data labeling.

Chapter 4

Value of ML Models

A few position papers have recently begun to challenge the assumptions that have driven the notion of quality in ML [122, 31] - namely the predominance of the notions of accuracy, precision, recall, and various measures of calibration errors. At the heart of this stance, there are two observations:

1. ML models are almost always applied with a default option, where the model can abstain or their inference can be rejected as shown in Figure 4.1.

2. The value (cost) of correct inferences, incorrect inferences, or rejections are in general not a property of the model but of the use case where the model is applied.

When we see things under this perspective, both the way we evaluate or select a model and the notion of what we consider to be ”learning” change.

The scenario of Figure 4.1 is central, not a corner case. This obviously applies to scenarios where errors are costly (medical diagnosis, self-driving cars): in those cases we ask humans to take control rather than making an inference that we are not sure of. But it is also the typical path in nearly any application: Siri and Alexa don’t always return the most likely action if they are unsure. This is also the norm in enterprise AI applications.

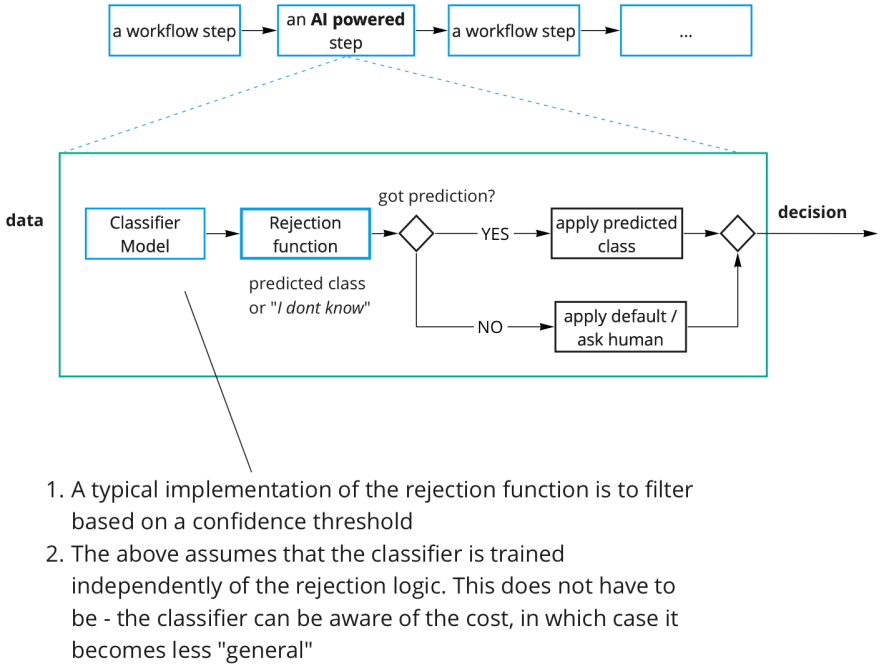


Figure 4.1: Typical implementation of ML models into an ML solution workflow - borrowed and edited from [122]

When we think in these terms, and if for simplicity we discuss the problem in terms of classification (though the concepts are identical for any AI capability), what we really mean to say is that our model is *applied as* a selective classifier [56]. Once we realize that selective AI models are the rule, not the exception, our approach to model evaluation changes:

An ML model provides a utility to each, and enterprises adopting a model may come up with some overall notion of utility, or value function. In the simplest case this value function could be accuracy or F1, but we have already seen that models are deployed as selective classifiers, and now we see that the value depends on the application use case, since for example the model being ”wrong” (or right) has different consequences.

1. The “value” of an “ML solution workflow” such as the one of Fig. 4.1, depends on how often the workflow rejects the predictions, on the correctness patterns of the predictions that the workflow lets through (not rejected),

and on the "cost" of errors vs the benefits of correct predictions. Notice that the value of the solution workflow depends on the use case (e.g., on how costly errors are), so that a workflow w_1 may be more valuable than w_2 for use case uc_1, but the reverse may be true for uc_2.

2. The value of a *model* therefore depends on the value of the "best" solution workflows we can build given that model. When we need to take a decision on which model to deploy given a set of options, value of the resulting solution workflow is the right driving factor.

This notion of quality is not what model accuracy, F1, or AUC measure. We may argue that accuracy metrics are a "good enough" proxy for data science-led model improvements and for model selection, and that all we need is to pick the model with the best accuracy and deploy it by filtering out predictions with confidence below a threshold, but in many cases this is wrong, both in how we select the model and how we integrate into the workflow, and can even unknowingly lead to negative utilities (meaning that we are better off without AI).

To some extent, all this is so trivial that it would not make sense to waste even a second of the reader's time. There is nothing special or difficult in having value functions, or in picking a model out of a set of models (or a set of training iterations) that performs well given a value function, or in assessing if a model performs well over a class of value functions. However, in this chapter, we show that if we accept the points above, then the method we use to measure, compare, and even train models change, and the implications of such changes are often neglected in the literature as well as in model leader-boards. In this chapter, we specifically make the following contributions:

- We show that universal metrics used for model evaluation can lead to wrong decisions.

- We show that the common approach to rejecting predictions (filtering by threshold) if applied naively - as it is often done based on our experience - leads to low or even negative model value and to significant loss of "value" with respect to what can be achieved.

- We show that the above is true even when calibration methods are applied - and briefly discuss why measures of calibration errors fail to capture the most important property of confidence scores, that is, the fact that the probability of confidence is high when predictions are correct (rather than the other way around).

- Finally, we show that simple, decades-old models, especially when trained in domain even with simple text encoders, behave very well and often better than large, complex models and provide some intuition of why that may be.

4.1 Related work

Mimicking the typical use of ML models in many practical applications, a number of approaches rely on the combination of a ML model making an initial prediction and a human annotator taking over when the model's confidence is not high enough [29]. Selective classifiers are specifically conceived for this use, by including a rejection mechanism to decide when to abstain from making a prediction. The literature on selective classifiers is quite extensive, covering a wide range of learning algorithms ranging from nearest-neighbour classifiers [62] to SVM [55] and neural networks [41, 45, 56]. The effectiveness of this solution is, however, heavily dependent on the reliability of machine confidence, which has shown to be very poor especially for deep learning [58, 17].

Hybrid Human-AI systems aim at solving classification problems by leveraging both humans and machines [108, 145]. Crowds have been extensively

employed to develop this type of systems, from learning crowd vote aggregation models from features of the crowd task [71], to leveraging crowds to learn features of ML models [35, 114].

Understanding the properties of a classifier is a critical step to effectively use it [68], and crucially relies on the notion of confidence for individual predictions. Various confidence-based techniques exist in the literature to detect those examples, such as using the entropy of the softmax predictions [133], measuring trust scores of classifiers based on the distance of samples to a calibration set [68], finding a confidence threshold (using either Shannon entropy [128], Gini coefficient [20], or norm-based methods [99]) that maximizes the coverage given a target accuracy [28], and using semantics-preserving data transformation to estimate confidence [15]. However, these confidence measures should be complemented with an appropriate value metric in order to assess the utility of the classifier in real-world applications.

4.2 Measuring model "value"

4.2.1 The setting

Selective classifiers can be implemented in several ways, such as:

(a) We take a model m that outputs a prediction p and a *confidence* c_p (or a confidence vector $\mathbf{c}$ with a confidence for a set of possible answers). Then, we filter the predictions to take only those above a certain confidence threshold (Fig. 4.2a).

(b) The model m outputs predictions (and possibly a confidence), but we apply a second model s (the selector) that decides whether to accept the prediction or not, based on some features of the input item i (Fig. 4.2b).

(c) A hybrid of the two above cases is where the selector is actually a recalibrator r that can either take as input only the prediction and confidence

measure or also the input features of i and adjust the confidence vector. We call the first as a *feature-agnostic* calibrator and the latter as a *feature-aware* calibrator (Fig. 4.2c).

(d) The model m is already trained to only output predictions that are "good enough" and includes an "I don't know" class (Fig. 4.2d).

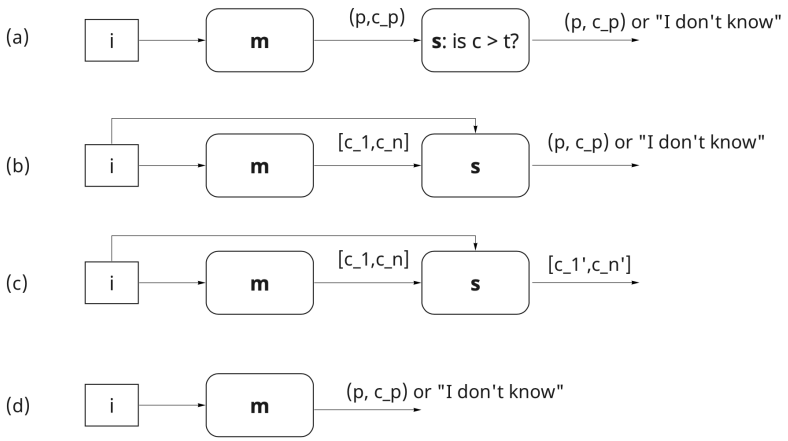


Figure 4.2: Typical implementations of selectivity in classification.

The first case is by far the most common, at least in our experience. The second case is an extension and generalization of the first case, in two ways: it can take features as input (that is, s can be trained as opposed to "just" being a formula), and it can filter based on any formula. Both the feature aware rejector and calibrator require some form of "training" or machine teaching. An important conceptual distinction here is that a feature-aware rejector makes sense in cases where we know the use case, because that knowledge will tell us when to reject. On the other hand, a feature-aware calibrator can be considered of general usefulness. However, if we consider feature-aware calibrator, one point we make is that in that case calibration and learning really are the same thing. In the end, we have a model $m' = r(m)$, and one could argue (we do) that calibration in this case is no different than learning or fine-tuning.

In formalizing "value", we will progressively make a few assumptions that i) allow to simplify the presentation of the problem without altering the essence of the concepts, ii) are reasonable in many if not most use cases, and iii) make the definition of the value function easier to understand and interpret for the users who eventually have to deploy ML into their companies. This is important: people understand accuracy because it is simple, and that has value even if accuracy is "inaccurate" as a metric, and most users will not be able to express complex value functions. We also scope the conversation on classification problems as it makes it easy to ground the examples and terminology, and because it is easier to define a notion of accuracy.

4.2.2 Definition of value

We have a classifier g that operates on items $x \in \mathcal{D}$ and returns either a predicted class $y \in \mathcal{Y}$ or a special label y_r, denoting "rejection" of the prediction. Given the above, we can compute the average value per prediction of applying a model g over $\mathcal{D}$ (so note that what we are talking about here is the value of a solution workflow). Specifically,

$$V(g, \mathcal{D}) = \rho V_r + (1 - \rho)(\alpha V_c + \Sigma_{ij}[\Omega \odot V_w]_{ij}) \tag{4.1}$$

where ρ is the proportion of items in $\mathcal{D}$ that are rejected by g (classified as y_r), α is the accuracy for predictions above threshold, V_r and V_c are the value of rejecting an item and classifying it correctly respectively, Ω is a matrix denoting the proportion of predictions (above threshold) in each cell of the confusion matrix, and V_w is a matrix with the cost for each type of error (set to zero on the main diagonal corresponding to correct predictions), and $\odot$ denotes the Hadamard (element-wise) product, of which we take the summation across all elements ij. Notice that ρ, α, Ω all depend on $\mathcal{D}$ and g, and we omit the indices to simplify notation. Also, if our classification problem has $|\mathcal{Y}|$ classes, then Ω and V_w are $|\mathcal{Y}| \times |\mathcal{Y}|$ (y_r is not included here). An alternative representation would be to

just say that $V(g, \mathcal{D}) = \Omega' V'$, where the confusion and value matrices incorporate the reject class. This would allow us to model the case where the value of rejections and of correct predictions is also class-dependent. Instead, if we only consider costs based on what we misclassify (based on the actual class) then Ω and V_w become vectors, and in the most common case where all wrong predictions are considered equally bad in a first approximation, then Ω and V_w are a scalar, and $\Omega = 1 - \alpha$, so in this case the formula simplifies to:

$$V(g, \mathcal{D}) = \rho V_r + (1 - \rho)(\alpha V_c + (1 - \alpha)V_w) \tag{4.2}$$

At this point, while we could carry on with this math, we simplify the notation for several reasons: the first is, well, to simplify the notation. The second is to remove dimensionality (values can be measured in dollars, but here we care about relative values because we want to compare models and learning strategies), and the third is to arrive at a formulation that is digestible for process owners (the people who apply AI in their processes), for whom it may be hard to come up with the three cost parameters/vectors. None of the above simplifications change the concepts presented.

Here we depart from [122] and define as baseline the case where we do not have ML, or, equivalently, we reject any prediction. We set this baseline at 0, which means that we set $V_r = 0$. This makes it easy for us to evaluate a model in terms of whether it improves on the baseline or not - and therefore in terms of whether we should adopt AI or not for a given problem.

$$V(g, \mathcal{D}) = (1 - \rho)(\alpha V_c + (1 - \alpha)V_w) \tag{4.3}$$

We also express V_w in terms of V_c, as in $V_w = -kV_c$, where k is a constant that tells us how "bad" is an error with respect to getting the correct prediction.

This leads us to:

$$V(g, \mathcal{D}) = V_c(1 - \rho)(\alpha - k(1 - \alpha)) \tag{4.4}$$

V_c is a scaling factor for the above value formula. When reasoning about an AI-powered solution workflow we do not really care about that factor, we can think in terms of value "per unit of V_c dollars", or equivalently assume magnitude of V_c, so we can focus on value. From now on we therefore focus on "value per dollar unit of rejection cost" $V' = V/V_c$. We avoid introducing a new symbol and, without loss of generality with respect to the above equations, we set $V_c = 1$ and get:

$$V(g, \mathcal{D}) = (1 - \rho)(\alpha - k(1 - \alpha)) \tag{4.5}$$

Notice that nothing really changes in the concepts we want to stress between equations 4.1 and 4.5, but the latter simplifies the presentation.

4.2.3 Filtering by threshold

We focus now on the most common situation observed in practice, the one in which the model selectivity is applied by thresholding confidence values and rejecting predictions that have confidence c less than a threshold τ (case (a) in Figure 4.2). In this setting, we are given a model m that processes items $x \in \mathcal{D}$ and returns a vector of confidences (one per class). Typically this is the output of a softmax. Specifically, for each x, we consider the pair $\hat{y}, \hat{c}$ corresponding to the top level prediction of $m(x)$ and the confidence associated with such prediction. Given a threshold t, we define a function s as:

$$s(\hat{y}, \hat{c}, t) = \begin{cases} \hat{y}, & \hat{c} \geq t, \\ y_r, & \text{otherwise.} \end{cases}$$

where y_r is a special class label denoting "rejection" of the prediction. Our classifier g is therefore now expressed in terms of m and t. This means that we can express the value as a function of $m, \mathcal{D}, t$.

In a given use case, when we are given a model m and have knowledge of Ω (or of k in the simplified case), we select the threshold $\tau \in [0, 1]$ that optimizes

$V(g, \mathcal{D})$ (We assume here τ is unique, or that we randomly pick one if not). This means that we can express the value of our classification logic as a function of $(m, \mathcal{D}, k)$

$$V(m, \mathcal{D}, k) = (1 - \rho_\tau)(\alpha_\tau - k(1 - \alpha_\tau)) \tag{4.6}$$

Notice that τ can be set empirically on some tuning dataset $\mathcal{D}$ (it depends on $m, \mathcal{D}, k$), and ρ_τ and α_τ reflect the proportions ρ and α given τ. However, if we are aware of properties of the confidence vectors, we can set τ regardless of $\mathcal{D}$. For example, if we assume perfect calibration (where the expected accuracy for a prediction of confidence c is c), then we know that the threshold is at the point where the value of accepting a prediction is greater than zero. If calibration is perfect, then $\alpha_\tau = \tau$. This means that to have $V(m, \mathcal{D}, k) > 0$ we need $\tau - k + k\tau > 0$, which means $\tau > k/(k + 1)$.

This conforms to intuition: if k is large, it never makes sense to predict, better go with the default. If k=0 (no cost for errors), we might as well always predict. If k=1 (errors are the mirror image of correct predictions), then our threshold is 0.5.

4.3 Experiments

There are many angles we can experiment with based on the concepts described. In this chapter, we explore:

- whether accuracy is indicative of model quality, and if a less accurate model may be preferable than a more accurate one, thereby implying that making decisions and determining leader-boards based on accuracy could be a limiting perspective at best.

- how to set confidence threshold based on value, and the extent to which calibration or threshold tuning affect value.

- which models and in which use cases perform well for different values of k

Specifically, we analyze both the behavior of simple as well as state of the art models over various datasets, models, and text encoders and provide insights on what model developers and process owners should look for in a model and in how to deploy it in a selective fashion. We refer the reader to our GitHub repo[1] for the companion code.

Task	Dataset(s)	Models	Model details
Hate speech recognition	Hate-speech detection	[14], [3]	Leader-board models
	Hate Speech&Offensive Language	LogR, MLP1, MLP4	from scikit-learn library
Clickbait recognition	Clickbait detection	fullnetconc, weNet, lingNet, fullNet	Leader-board models
Sentiment analysis	Multi-Domain Sentiment Analysis (MDS)	mttri [116]	Leader-board
		Google's T5-base	fine-tuned for sentiment analysis
		SieBERT [61]	fine-tuned RoBERTa-large [88]
		LogR, MLP1, MLP4	from scikit-learn library
		GPT-3	we fine-tuned for binary sentiment analysis
	Twitter US Airline Sentiment	LogR, MLP1, MLP4	from scikit-learn library
	Coronavirus tweets NLP	LogR, MLP1, MLP4	from scikit-learn library
Content classification	DBPedia Classes	LogR, MLP1, MLP4	from scikit-learn library
	Yelp-5	LogR, MLP1, MLP4	from scikit-learn library
	News Category Dataset	LogR, MLP1, MLP4	from scikit-learn library
Intent classification	Clinc150	LogR, MLP1, MLP4	from scikit-learn library

Table 4.1: Tasks, datasets and models used in the experiments

4.3.1 Experimental Setup

Tasks, Datasets and Leaderboards We experimented on a set of text classification tasks where making errors is especially harmful. In Table 4.1 - and more in detail in the appendix, Table A.1 and Algorithm 3 - we list the tasks, datasets, and the algorithm experimented with for each task and dataset.

Hate-speech detection on Twitter. [6] analyzed two widely used models ([3, 14]) and tested on popular twitter hate-speech datasets ([142, 43, 157]) with different settings.We replicated the original tests of the two models in *Ex-*

[1] https://tinyurl.com/rethinking-value-of-ml-models

periment 1 and then analyzed their performance under the settings by [6] in *Experiment 2* (more details in the appendix A.1.3)

Clickbait detection. The *Clickbait Challenge* on the *Webis Clickbait Corpus 2017*[2] was classifying Twitter posts as a clickbait or not. Both training and test sets are publicly available[3], while each team was free to choose a subset of the training set for validation (we followed the "blobfish" team).

Multi-Domain Sentiment Analysis - and Dataset (MDS). Sentiment analysis based on a publicly available dataset for domain adaptation[4]. The data includes four categories of Amazon products (DVD, Books, Electronics, and Kitchen), and the task is to learn from one of these domains and analyze the sentiment on the others.

In addition to the above binary classification problems and data, we used seven publicly available multi-class datasets with different class distributions (see Table A.1 in the appendix for details).

Models. We used various models for each task in our experiments (see Table 4.1 and appendix A.1.2 for the details). For the Hate Speech and Clickbait datasets, we tested the leader-board models. For MDS dataset, we used the leader-board model "*Multi-task tri-training (mttri)*" by [116], two transformer models (a T5-base model fine-tuned for sentiment analysis[5] and SieBERT [61], a fine-tuned checkpoint of RoBERTa-large [88]), as well as a simple Logistic Regression (LogR) model and two multi-layer perceptron models from the scikit-learn library[6] with one (MLP1) and four (MLP4) hidden layers respectively. We used LogR, MLP1 and MLP4 for the multi-class datasets. We tested simple models with different text encoders: (i) *TF-IDF*, (ii) *MPNET*, and (iii) *nnlm* (details in A.1.4), but for simplicity we show the results with *TF-IDF* (see the appendix-Figure A.11 for further results).

[2]https://webis.de/data/webis-clickbait-17.html
[3]https://zenodo.org/record/5530410#.YWcFtC8RrRV
[4]http://nlpprogress.com/english/domain_adaptation.html
[5]https://tinyurl.com/t5-base-finetuned-sentiment
[6]https://scikit-learn.org/

Cost Settings. Following the simplification in Section 4.2.2, we set $V_r = 0$ and $V_c = 1$, and then test the models using different values of $k \in [0, 10]$). In binary tasks, we consider the cost of false positives (k_{fp}) and false negatives (k_{fn}) separately.

4.3.2 Results

Accuracy vs Value

We first investigated the extent to which models are robust across varying cost factors k, and consequently also whether we can use accuracy metrics to select the "best" model to deploy in an ML platform. We did so both for challenges/ leader-board models and for the set of small/simple and larger models as described. As an example, Table 4.2 shows results on MDS dataset that even for fairly small and very realistic cost factors, the model we would choose with a value oriented approach differ from what we would choose based on accuracy. The appendix show many other cases where this happens - as well as cases where instead the model with the best accuracy also has best value across several costs metrics. Notice that $k = 4$ is actually a very small and realistic cost factor: it means that "being wrong is 4 times as bad" with respect to the advantage of being right. Most scenarios have values of k way more extreme. Notice also that accuracy corresponds to the case where we do not reject any predictions. This is equivalent to setting $k = 0$. Indeed, not filtering examples (accepting even low confidence predictions) means that we do not care about being wrong. Another important observation is that in many cases, across models and datasets, we often find that even leader-board models have *negative value*, and even for cost factors of $k = 1$.

MODEL	ACCURACY	VALUE				
		K=1	K=2	K=4	K=8	K=10
LOGREG	0.762	0.524	0.339	0.162	0.053	0.033
MLP1	0.749	0.497	0.327	0.18	**0.081**	**0.062**
MLP4	0.735	0.47	0.24	-0.143	-0.78	-1.06
MTTRI	0.808	0.616	0.441	0.148	-0.354	-0.58
T5	0.784	0.568	0.352	-0.08	-0.944	-1.376
SIEBERT	**0.842**	**0.685**	**0.527**	**0.217**	-0.397	-0.705

Table 4.2: Performance of models on MDS Dataset, TARGET = Electronics (using theoretical threshold)

Calibration vs Threshold Optimization

As explained in Section 4.2.3, we expect that the theoretical threshold would maximize the value of a model if it is well-calibrated. If this assumption is not true, then we should either find an "optimal" threshold empirically by tuning it on a validation set, or we should first calibrate the model (e.g., via temperature scaling [58] or other methods) and then maximize expected value by filtering based on the theoretical threshold. We compared how accuracy and value are affected by either calibrating the model or tuning the threshold on a "validation" dataset. For calibration, we first calibrate models via temperature scaling and then use the theoretical threshold to compute the values. For tuning, we investigated how empirically choosing the confidence threshold for each $(model, task)$ pair affects value. We used a validation set to find the threshold that maximizes the model's value for every single k, and then used those thresholds to compute the values on test set.

Figure 4.3 shows the results for uncalibrated models, calibrated models (second column) and threshold tuning (third column) for the hate speech and clickbait cases (and again more results in the appendix and additional material). Notice that calibration helps but still leads to very low, zero, and sometime negative values even at low cost factors (this is the case for Hate speech dataset when we

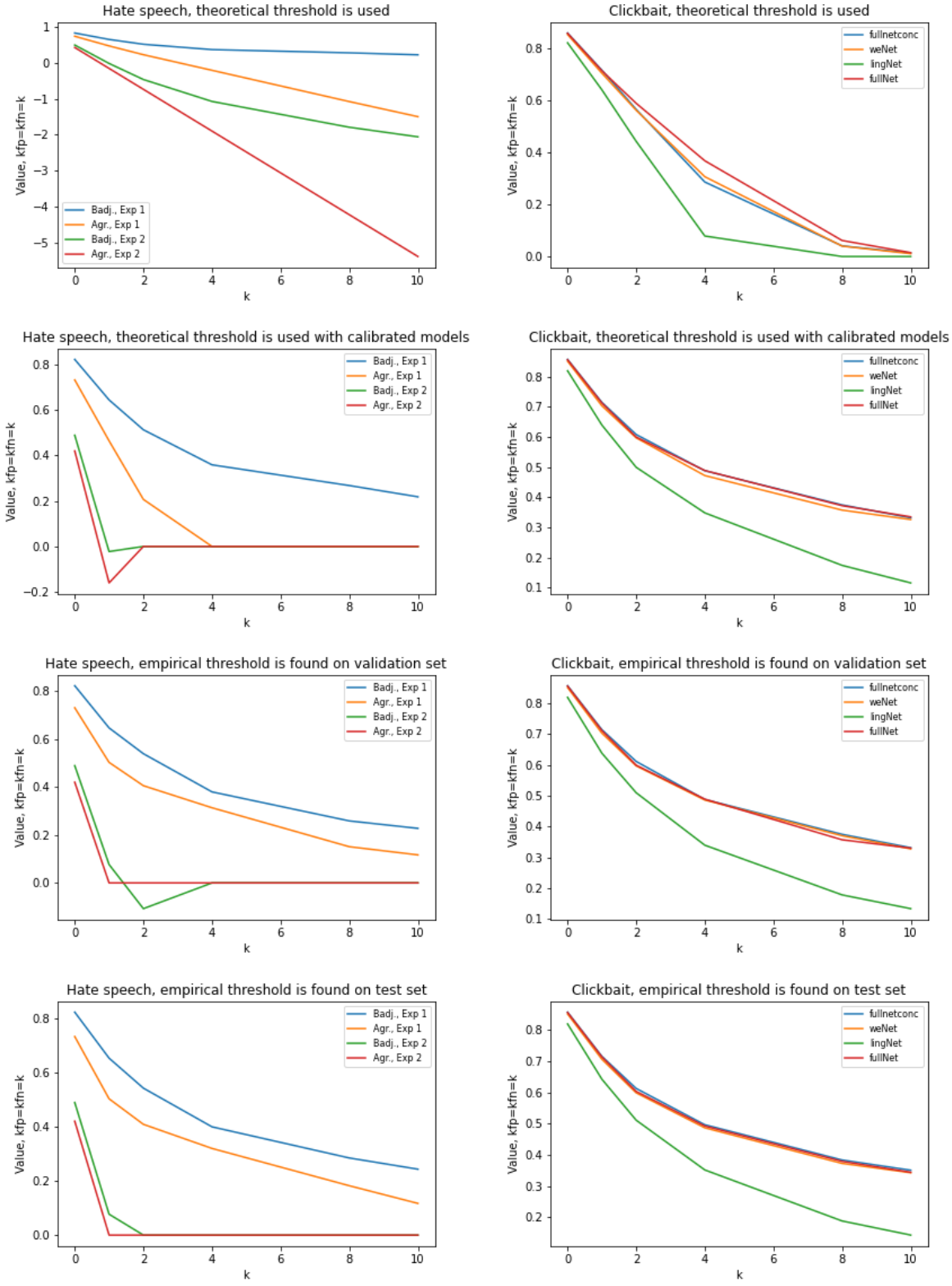


Figure 4.3: Value curves of binary datasets for increasing k.

run Experiment 2 -see Figure 4.3, second column). The empirical threshold found on the validation set provides equal or better values than the theoretical threshold in almost all cases.

The rightmost column of Figure 4.3 reports the highest achievable results, obtained by optimizing the threshold on the test set itself, showing how tuning the threshold on the validation set is close to optimal in most cases. Still, there are cases (*Hate-speech*, Exp. 2) in which the models are useless (Value=0, all predictions rejected) for most values of k. We report results on all the datasets in the Appendix A.1.6 and show that they are consistent with these findings.

MODEL	TARGET			
	DVD	BOOKS	ELECTRONICS	KITCHEN
LOGREG	0.74	0.704	0.762	0.782
MLP1	0.728	0.691	0.749	0.765
MLP4	0.72	0.696	0.735	0.761
MTTRI	0.753	0.742	0.808	0.821
T5	0.788	0.770	0.783	0.777
SIEBERT	**0.836**	**0.826**	**0.842**	**0.865**

Table 4.3: Average accuracy of models on MDS Dataset.

The effect of complexity and out-of-distribution data.

We investigated why models that rank high in terms of accuracy drop in quality when the cost ratio increased, while others are more ”robust. As an example, we show results from an experiment on the *MDS* dataset to see the impact on cross-over domains (on out-of-distribution samples). The leaderboard model for this task is ”*mttri*” [116] but we also tested two transformer models (a fine-tuned *T5-base* and *SieBERT*), LogR, MLP1 and MLP4 as explained in Section 4.3.1. We trained all the models (except T5 and SieBERT) on each domain and tested on the other 3 domains separately (so that we have 12 different cases of <source domain,target domain> pairs). We then calculated the average values of each model on each target domain (see Table 4.3 for the average accuracy of

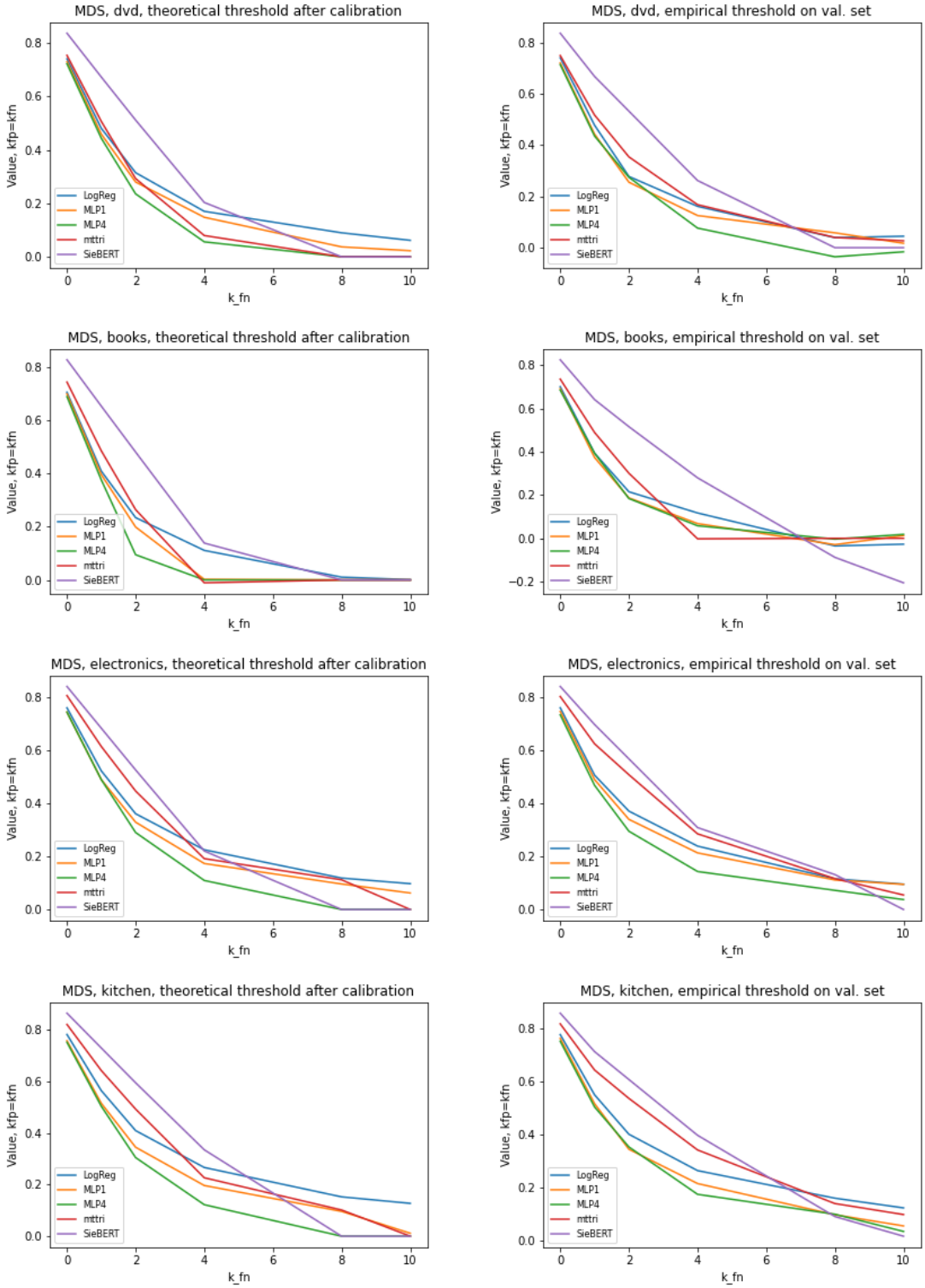
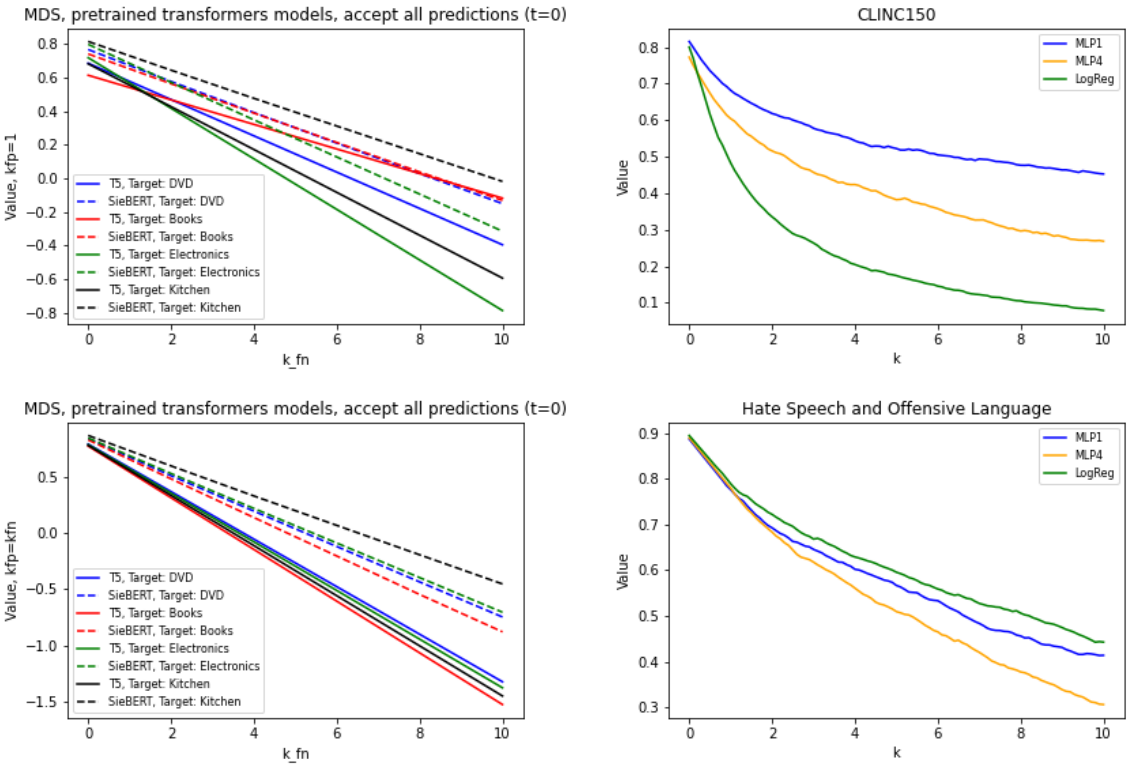


Figure 4.4: Value curves on MDS dataset for increasing k, using theoretical threshold after calibration (left column) and empirical threshold on validation set (right column). Values are averaged except the 'SieBERT' model.

each model). *SieBERT* is the best one (even better than the leader-board *mttri* model) based on accuracy, but this is not the case in terms of value.

Figure 4.4 shows the results. In most of the cases simple LogR and MLP1 models with simple text encoders (i.e. *TF-IDF*) have better value than the complex models for high k values. We also tested a fine-tuned version of $T5 - base$ model for sentiment analysis (which does not output the confidence values associated to each prediction), that is why we could only measure its value by accepting all the predictions (see Figure 4.5a). Related to this, notice how the inability to filter (reject) predictions lead to negative value, even with the fine-tuned T5 that performs very well in terms of accuracy.



(a) Transformer models on MDS dataset, no threshold.

(b) Multi-class datasets, simple models with theoretical threshold

Figure 4.5: Value curves for increasing k.

We repeated the experiments also on seven multi-class datasets and observed

that for example the MLP4 model is worse than a simple MLP1 model over almost all datasets. Furthermore, MLP4 is even worse than a simple Logistic Regression model on four datasets (please see Figure 4.5b for an example for each case, and the rest can be found in the Appendix-Figure A.9).

The pattern in the experiments seem to suggest that simple models (usually thought to be naturally well calibrated) perform relatively well when errors are costly, and that for high cost factors models trained on a different domain also tend to perform poorly, which suggests that even simple models trained on domain can be preferable. This observation is crucial in enterprise AI, where each company and each vertical has its own ”language” skew. Interestingly and perhaps not surprisingly, large pretrained language models that are not bottlenecked by insufficient training data perform well across the board. This can be due to two reasons (besides the models being very powerful): first, we know that large models with very large train datasets are reasonably well calibrated (e.g. see [69]). Second, when the training data is so large, fewer examples are out of distribution in terms of language. For example, GPT-3[7] is trained on about 45TB of text data from various datasets and it performs very well on MDS dataset (see Table 4.4 - and the appendix A.1.5 and Figure A.3). The reason is that the MDS dataset is quite old and probably GPT-3 model has already learned it (so MDS dataset does not include out-of-distribution samples for GPT-3 model). However, such models may be too costly or impractical due to their scale, and the problem still remains for enterprise datasets which may be quite different from what the large pretrained model has seen. Furthermore, as the cost grows, the difference with respect to simple models drop significantly.

[7] https://openai.com/api/

TARGET	ACCURACY	VALUE				
		K=1	K=2	K=4	K=8	K=10
DVD	0.832	0.664	0.534	0.367	0.164	0.089
BOOKS	0.806	0.613	0.46	0.272	0.077	0.004
ELECTRONICS	0.82	0.641	0.499	0.322	0.127	0.051
KITCHEN	0.853	0.706	0.599	0.464	0.308	0.251

Table 4.4: Performance of fine-tuned GPT-3 on MDS Dataset (Average values using theoretical threshold)

4.4 Limitations and Conclusion

The takeaway from our experiments is that using accuracy-oriented metrics (that is, metrics that assume models are applied without rejection) is as a minimum a risky proposition - and this is true even for models widely acknowledged as "leaders". We should always assess models over a range of cost factors, and at least for reasonable cost factors we expect based on the set of application use cases we are targeting. $k = 0$ (accuracy) is almost never a reasonable one. We also saw how applying models without thresholding can lead to negative value, and that threshold tuning seems to perform better than calibration. We also hypothesize and have obtained some support for identifying complexity and out-of-distribution as factors that may lead to rapid model quality degradation for higher cost factors.

This being said, we see this work more as providing evidence of a problem and outlining the research needs: more studies (especially with large models and in vs out of distribution datasets) are needed to validate the hypothesis and a deeper understanding of how calibration, confidence distribution, and size of validation set affect model value.

Chapter 5

Active learning

Despite remarkable advances in ML, training data remains a key bottleneck for the successful application of ML techniques. Obtaining a large amount of high-quality training data is usually a long, laborious, and costly process. AL provides an effective means to accelerate the process, by iterating data labeling and model training, and identifying at each iteration which data to label next, to converge more rapidly and effectively to an accurate model. *Crowdsourcing* is often used in conjunction with ML, both as a way to collect labeled data efficiently and as a way to assist trained models for predictions where the model confidence is not deemed sufficient ([29, 77]). Despite their joint usage, the interaction between AL and crowdsourcing has been largely unexplored. This interaction is non-trivial for many reasons: for example, crowdsourcing typically produces rather noisy labels and the impact of such noise on ML algorithm confidence estimation and calibration is still unclear. Furthermore, at every AL iteration, we are faced with several choices, from how to aggregate crowd votes on a label to whether we should ask the crowd to label new data items or verify (reduce the noise on) already labeled items - and these choices may impact AL performance.

This chapter reviews existing AL approaches and investigates their performance in the hybrid human-machine classification setting, where crowd work-

ers contribute labels (often noisy) to either directly classify data instances or to train an ML model for classification. Unlike existing surveys ([125, 2]) that focus on the algorithmic design of AL and the review paper about integrating AL with deep learning ([27]), here we aim at re-evaluating existing AL approaches in terms of cost-effectiveness when data labels are crowdsourced, and so given the constraints of a limited crowdsourcing budget and noisiness of worker-contributed labels.

To this end, we first review existing AL approaches under three categories, based on their reliance on a *strategy*, that is, specific queries to get the sample of interest from the data. These categories are: (i) *fixed-strategy approaches*, that apply a specific item selection method regardless of the data or problem, (ii) *dynamic-strategy* approaches that have a portfolio of strategies and choose one each time they need to sample a batch of items for labeling, based on past performance on that specific problem and data, (iii) *strategy-free* approaches that do not have any apriori selected portfolio of strategies, but rather learn the best strategy from scratch based on the problem, data, and prior experiences. We also review proposals in the literature that discuss how AL can deal with noisy labels. We then report the results of an extensive experimental comparison evaluating the performance of the different AL approaches in human-machine classification.

We evaluate the performance of AL approaches under two different scenarios: 1) *ML only*: this is the "traditional" approach of training a model with AL and then testing its performance on a pool of items. 2) *Hybrid*, where crowd and ML interact also in the classification phase, not just in data labeling. Indeed, many problems we face have a *finite pool*, where the set of items to classify is finite, and there is, therefore, a trade-off between spending our budget or effort to train an ML model (using AL methods) versus spending that budget to directly classify items in the pool via the crowd, or using a combination of crowd and ML.

To run this comparison, we developed a library of AL approaches collecting implementations provided by the authors when available, and re-implementing them when we could not find existing code. As part of this process, we also created a collection of crowdsourced datasets containing micro-level information (i.e., individual crowd votes), by aggregating the publicly available ones and adding the ones we collected (note that most available crowdsourced datasets do not make individual votes available). Both the software library [1] and the collection of benchmarking datasets [2] are made freely available to the scientific community. We believe that both the implementations and the crowdsourced micro-data will be an important contribution in their own right given the difficulty we had in obtaining both, despite extensive searches.

An important lesson learned from the experimentation is that prior conclusions on the performance of AL approaches obtained in non-crowd labeling settings cannot be blindly extended to crowdsourced data. Specifically, strategy-free approaches that have shown to be effective in many contexts do not achieve the best performances across crowdsourced datasets. We speculate that this can be due to the impact of noise in ML model calibration and uncertainty estimation. We also observed that hybrid classification improves the performance of AL approaches over crowdsourced datasets.

In summary, we make the following contributions:

- We identify three categories of AL approaches in the literature and analyze their characteristics and effectiveness.
- We contribute a library of implementations of state-of-the-art AL algorithms and a collection of benchmarking datasets for human-machine classification.
- We report the results of an extensive experimental evaluation, providing

[1] https://tinyurl.com/source-code-data-results

[2] https://github.com/TrentoCrowdAI/crowdsourced-datasets

insights into the performance of existing AL strategies in hybrid human-machine classification contexts.

- We provide a critical discussion on the main insights that emerged from our analysis, highlighting relevant open challenges and potential future directions to address them.

5.1 Active Learning Strategies: A Review

AL ([39]) has been a very lively research field over the last decade. Given a set of items I and an ML algorithm M, AL aims at defining a strategy to progressively sample items from I on which to obtain true labels, so that M can be trained with a smaller dataset with respect to random item sampling. The underlying assumption is that obtaining training data is costly, and therefore minimizing the size of such dataset for a given target accuracy is highly beneficial ([70]). In the following, we review existing AL approaches by grouping them in terms of the type of strategy used to choose the sample of interest.

5.1.1 Fixed-Strategy Approaches

Pioneering fixed-strategy approaches have been proposed in the 1990s. [127] proposed the *query by committee (QBC)* approach, which polls a committee of different classifiers trained on the current set of labeled items to predict the label of each unlabeled item. Then, items to label are selected based on the maximum degree of disagreement among the classifiers. They showed that the prediction error decreases exponentially fast in the number of queries. The approach and experimentation are however limited to parametric learning models with continuously varying weights and cases where learning is perfectly realizable, and the learning algorithm is the Gibbs algorithm ([60]). In [54], they proved that *QBC* such an exponential decrease is guaranteed for a general class of learning prob-

lems. They used two machine classifiers as the "committee", and determined general bounds on both the number of queries and the number of instances to be labeled. Specifically, the paper defines higher and lower bounds for the expected information gain of *QBC* and proves that if the queries have a high expected information gain then the prediction error is guaranteed to decrease rapidly with the number of queries. [94] follow a similar "committee-based" approach but apply *expectation-maximization (EM)* in to determine class probabilities and the extent to which classifiers disagree, and weigh item selection by their *density*, defined as the distance from a document to the others. With this approach, they reduced the required number of labeled documents by 42% over the previous *QBC* approaches.

[81] presented the idea of *uncertainty sampling*, where the intuition is to sample items on which a model M is more uncertain, and this approach has long been a de-facto standard in the AL literature. They showed that aiming at reducing the uncertainty of M significantly decreases the number of items that must be labeled to achieve the target accuracy. [38] proposed *selective sampling*, based on the idea of identifying uncertain regions in a vector space used to represent items and then selecting items (points in space) from such uncertain regions to minimize them. In the case of support vector machines, uncertainty sampling is implemented by selecting instances that are closest to the decision boundary ([135]).

[115] introduced the *error-reduction sampling* approach, aiming at selecting items that will reduce the expected error of the active learner in the next test examples. They computed the expected error rate of an item either by using the entropy of the posterior class distribution (log-loss) or by using the posterior probability of the most likely class (0-1 loss). [96] focused on the same objective and proposed the *MinExpError* approach that uses the theory of non-parametric bootstrap ([49]) to design generic and scalable sampling strategies. First, bootstraps are created and assigned to different classifiers; then, the ex-

pected error of these classifiers for every single item in the unlabeled data is computed; finally, the items that minimize the expected error are selected. The authors showed that the *MinExpError* algorithm requires significantly fewer labeled items than existing approaches back then.

Probabilistic Active Learning (PAL) ([47]) combined the idea of uncertainty sampling and the expected error reduction with smoothness assumption ([32]). The underlying assumption is that if two items are close in the feature space, then their labels should also be close. An item is represented by two attributes; (i) the total number of labeled instances in the neighborhood of the item, and (ii) the posterior estimate for the total number of the positive labeled neighbor set. This approach uses probabilistic estimates to investigate the neighborhood statistics of an item (label statistics) and measures the overall gain in classification performance (probabilistic gain) in terms of a user-defined point classification performance ([105]). It then selects items that improve the expected probabilistic gain most within their neighborhood. Its time complexity is comparable to uncertainty sampling, and it provides fast and stable performance.

[118] proposed the *Bootstrap-LV* approach, which detects the variance in the probability estimates of bootstrap samples and uses weighted sampling to find the most informative items. Another weighted-sampling approach is known as *Importance-Weighted Active Learning (IWAL)* ([21]) which applies an adaptive rejection sampling to each instance and assigns an importance weight (the inverse probability of being retained) to each retained item. [22] improved *IWAL* by using a rejection threshold based on the importance-weighted error estimates that minimize the prediction error. They showed that this approach improves the label complexity which reflects the intrinsic difficulty of the learning problem ([140, 150]).

Additional strategies include clustering instances and selecting cluster representatives as the most informative items ([26]), or combining representativeness and informativeness of instances to minimize the maximum possible classifica-

tion loss (i.e. *Query Informative and Representative Examples (QUIRE)* ([65])).

Although many of these approaches explicitly target generalization performance improvement, by means of expected error reduction ([115, 163, 59, 96]), or variance reduction ([124, 126, 63]), fixed-strategy approaches are unlikely to work on all scenarios ([18, 64]). The reason is that they rely on intuitions and heuristics that do not generalize to all datasets and ML problems. For example, even in our experiments, discussed next, uncertainty sampling performs well when false positives and false negatives have the same ”cost”, but less so when errors, and specifically errors of a specific type are more costly than others. This reveals that fixed-strategy approaches are not able to adapt to the data and the problem at hand.

5.1.2 Dynamic-Strategy Approaches

Approaches to dynamic strategy selection are in essence based on progressively learning which AL approach works best for the data at hand. This kind of “learning to learn” approach was first proposed by [18], who showed that one single strategy cannot perform well on all problems. Their approach, named *COMB*, combines a group of AL strategies and dynamically evaluates them to achieve the best possible performance on the problem at hand. Although the online selection of strategies expedites the AL process, combining multiple strategies and evaluating their individual performance brings two challenges ([18]). First, as dynamic strategies choose the next action by estimating the performance of each AL approach given the current state and the past observations, the quality of such estimation becomes crucial. However, items selected by the active learner are biased to be the “hard” ones and do not reflect the exact distribution of the items, and as such the quality estimation in absence of a test dataset (which is rarely available in AL) is biased. Second, at each batch only the label of the instances proposed by the selected AL approach are available; there is no way to know the consequences of labeling other instances proposed

by other strategies. To handle these challenges, *COMB* ([18]) is designed as an adversarial multi-armed bandit problem (*MAB*) ([9, 10, 8]) combined with the *EXP4* algorithm ([10]), where AL strategies are considered as "experts" and unlabeled items are the "slot machines". Thus, the selection of an item is based on the opinion of all experts.

[64] proposed the *Active Learning by Learning (ALBL)* algorithm as an extension of *COMB*. It represents each bandit machine as an AL approach, and uses the *EXP4.P* algorithm ([23]) to select a machine adaptively. The main differences between *COMB* and *ALBL* are as follows: (i) while *COMB* represents each machine as a single unlabeled instance, a machine corresponds to an AL approach in *ALBL*, (ii) both *COMB* and *ALBL* adopt the EXP4 algorithm, but *COMB* restricts each machine to being pulled only once, while a machine can be pulled many times in *ALBL*, and (iii) *COMB* uses human-designed evaluation criteria based on entropy, while *ALBL* uses an unbiased estimator of the test accuracy (weighted accuracy) to decide the rewards of the single strategies. [64] showed that *ALBL* gives either comparable or better results than *COMB*. In general, the above papers show that ALBL works better than fixed-strategy approaches when the problem is easier to learn, while it is comparable for harder problems (where strategy selection is also more challenging).

While *ALBL* probabilistically blends the items suggested by different AL strategies to select the most informative one, [37] proposed blending the strategies themselves to build an aggregated strategy. Their approach, called *LSA* (Linear Strategy Aggregation), combines *LinUCB* (linear upper confidence bound) ([82]), a state of the art MAB approach, with the task at hand. They represent experience as the weights with which to aggregate strategies, and adaptively adjust these weights when tackling a new problem. They aim to transfer this experience learned from the model to other AL tasks through biased regularization. They proved that the transfer of the learned experience is beneficial to achieve better performance.

Although dynamic-strategy approaches use bandits to ensemble multiple strategies in the learning process ([18, 64, 37]), they still assume that there is a single best combination in each batch (stationary bandits). In so doing they are not robust to non-stationary cases, where the weighting proportions must be adapted over time in the learning process. Recently, strategy-free approaches have been proposed to overcome these limitations.

5.1.3 Strategy-Free Approaches

Strategy-free AL processes use prior experience (meta-data) to learn new tasks (*active meta learning*). The main difference between *dynamic* and *strategy-free* approaches is that the latter does not rely on any human-designed strategies.

In this category, [72] devised a novel data-driven AL algorithm, named *Learning Active Learning (LAL). LAL* is formulated as a regression problem that learns how to predict the reduction in the expected generalization error when we add a new label to the training set. It uses Monte-Carlo sampling to correlate the test performance directly with the classifier and item properties. Both the classifier and the items are represented with a set of parameters so that *LAL* can sense any change in the training set. As a result, the learning state is continuously tracked as a vector whose elements depend on the state of the current classifier and on the selected item. A drawback of this algorithm is being classifier-specific, that is designed as a random forest regressor.

[146] and [51] use reinforcement learning (RL) for learning an active learner in a data-driven approach. They adopted a stream-based AL process in which the agent observes the data in sequence and decides whether a single item should be labeled by the agent itself or it should be asked to an oracle. Based on the decision, the agent receives a reward and a prediction model is adopted to be used in new tasks. They improved the performance of models, but they tend to learn only from related datasets and domains ([73].

Many other data-driven approaches for pool-based AL processes have been

proposed recently. While [13] and [103] used RL to build the learning model, [86] formulated learning AL strategies as an *imitation learning* problem (i.e., the machine is trained to perform a task from demonstrations by learning a mapping between observations and actions), [40] and [111] applied *few-shot learning* (i.e. classifying a new data having seen only a few training examples), and [102] extended the *LSA* approach using non-stationary multi-armed bandit with expert advice.

Active meta-learning approaches have also been developed in application-specific scenarios. [130] developed (*ActivMetaL*), an active meta-learning recommender system. *ActivMetaL* keeps the scores of multiple AL approaches on given tasks in a sparsely populated collaborative matrix, predicts the performance of each approach for a new task, and then fills the corresponding row of the matrix for this task. In this way, they predict which algorithm will perform best for the new task/dataset.

While capable of generalizing across learning tasks, these meta-learning approaches still have many limitations, such as being classifier specific ([13, 40, 111]), having a greedy approach by missing a long-term reward ([86]), or being limited to specific domains (i.e. imitation learning, or few-shot learning) ([13, 51, 86, 130]).

To overcome these limitations, [73] proposed an RL variant of their *LAL* algorithm, named *LAL-RL*, that defines AL as a Markov Decision Process and tries to find the optimal and general-purpose strategy. *LAL-RL* is independent of dataset and ML classifier (contrarily to *LAL* that is designed for Random Forests), and its objective does not depend on a specific performance measure. The authors show how *LAL-RL* can transfer learned strategies across substantially different datasets.

Recently, [48] tested the performance of *LAL-RL* on 20 real-world datasets and compared it to *random sampling* and *uncertainty sampling*. Although *LAL-RL* shows very good performance on average, it is not always better than ran-

dom sampling, especially in the case of highly unbalanced datasets ([48]). In addition, their results show that the choice of the model is decisive (i.e. random forest classifier gives better results than logistic regression). They also report that *LAL-RL* is sensitive to the metric used to evaluate the performance, and it requires optimizing many hyper-parameters. This analysis shows that even general-purpose strategy-free approaches have limitations when dealing with real-world problems.

Although the main objective of these meta-learning approaches is to adapt the prediction/learning model to new environments/tasks, they do not consider the characteristics of the target environment in the prediction model. Hence, they are likely to be effective in similar environments only. [139] proposed a new approach that learns a good policy directly based on the target environment either by using a pre-trained AL model or learning a new policy from scratch considering the budget for human annotation. They showed that this approach is more effective than the previous work ([51, 86]) when the source task and the target task are different. [117] proposed a deep Q-learning approach that is capable of dealing with multiple environments by learning a multi-modal AL strategy. They focus on the task of engagement estimation from real-world child-robot interactions during autism therapy. They employ an LSTM network to classify the individual modalities into engagement levels (i.e. low, medium, or high) and feed its predictions into the deep RL agent, making it capable of efficiently personalizing the interaction strategy to the target user.

In summary, the state of the art shows that the existing *active meta learning* approaches can outperform the fixed-strategy and strategy-free approaches, especially when the dataset is not highly unbalanced. However, most of them do not present comprehensive benchmarks to prove the transferability of the learned policies into real-world tasks (i.e. when we do not have a separate test set) ([48]). While showing a noticeable improvement in terms of generalization ability with respect to previous approaches, meta-learning approaches still

cannot cope with all challenges that are to be faced in real-world scenarios.

Specifically, nearly any real-world application we encountered has two aspects that haven't received much attention so far in AL research: the first is that the amount of noise in the labels is much higher with respect to standard settings where labels are provided by domain experts, and in crowdsourcing, there are several trade-offs we can make to reduce the noise and to balance budget vs noise trade-offs (for example, we can collect more votes for the same items and aggregate them to get a more reliable label, or we can pay more budget to ask for highly rated workers, or we can instead focus on getting a larger number of items labeled at low cost, although with higher noise). The second is that the cost of false positives and false negatives (and more generally of different types of errors) is rarely the same and that low calibration error is often a key quality of a good ML model. There are however a few contributions to dealing with noisy labels and we discuss them next.

5.1.4 Dealing with Noisy Labels

While crowdsourced labels can be made to be very precise via redundancy and aggressive worker selection/crowd testing strategies, the individual votes are often noisy. There indeed exists a line of work in the AL community that explicitly focuses on dealing with noisy labels. [159] proposed combining uncertainty and inconsistency (entropy of the label distribution) measures to select instances. The underlying idea is that the learning strategy should select items that are in unexplored regions or near the ones that may have been mislabeled. They also propose relabeling the mislabeled items via crowdsourcing. They show that when the labels are noisy and the aggregated label is not trustful (i.e. the aggregated label is provided by less than 50% of the workers, who annotated the corresponding item, in a binary classification setting), then relabeling significantly improves the performance of AL. [24] proposed a method for identifying and mitigating mislabeling errors, where they derive an infor-

mativeness measure to see how much a queried label would be useful if it was corrected. They show that this approach is more efficient in characterizing label noise compared to the commonly used entropy measure. Then, [25] extended this approach by measuring how much the queried item's label is likely to be wrong, based on disagreement with the current classification model, without relying on crowdsourcing.

A related line of work aims at addressing label noise by explicitly modeling the uncertainty of annotators. [152] introduce a model that jointly learns a classifier and infers annotators reliability, in an active learning setting that involves both data instance and annotators selection. [52] consider the case when annotators can learn from one another to improve their annotation reliability. [161] model a scenario where workers can explicitly express their annotation confidence, by allowing them to choose an unsure option. [151] further consider labeler properties such as consistency. [153] extend the problem to enabling deep active learning from crowds, i.e., enabling deep neural networks to actively learn from crowd workers. However, none of them tried to actively estimate the noise level of data during the learning process.

5.2 Experimental Work

As we have seen, the near totality of existing AL approaches i) assume that oracles provide the gold (ground truth) labels, ii) strive for accuracy as a metric, and iii) assume classification is done by ML only. In reality, the situation and needs are different, especially when we have access to a large set of human annotators of different reliability. First, labels can be noisy; second, the trade-off between the benefit of correct classification and the cost of an error varies greatly by application, and achieving a low calibration error may be as important as achieving high accuracy; finally, in many scenarios, we do have the option of relying on human classification when ML is uncertain, and in those contexts,

ML must learn to know when it doesn't know.

In this section, we examine the behavior of AL approaches in crowdsourcing settings. Specifically, we focus on problems where we start from a blank slate, have a pool of items to classify and a crowd at our disposal, and need not only to choose/assess AL approaches but also to assess if the crowd is leveraged only to get labeled data for training or also to perform classification at inference time, as done in hybrid classification contexts ([75, 29]).

5.2.1 Problem Formulation

We focus on hybrid binary classification problems where classification is accomplished via the combined contribution of humans and machines. The problem can be formulated as follows. We are given a tuple of (I, M, Q, B), where:

- I is a pool of unlabeled items to be classified.
- M is an untrained ML classifier.
- Q is an AL query strategy.
- B is the budget available (expressed in terms of total number of crowd votes we can ask).

Our aim is to classify all items I via the ML classifier M or/and crowd workers, assuming we do not have any training data to start with. To achieve this, we apply AL strategy Q for querying the most informative items $T \subset I$ that will be annotated by crowd workers on money B. The annotated items T will be used as training data for machine M and finally I will be classified based on M and the feedback collected from the crowd. Our hybrid AL workflow is described in detail in Algorithm 1.

Algorithm 1 Hybrid AL workflow

Input: I, M, Q, B

$LI, UI \leftarrow \{\}, I$ — *labeled and unlabeled items*
$T \leftarrow \{\}$ — *training dataset, {(item, label), ..}*
$VotesAll \leftarrow \{\}$ — *set of crowd votes on items, {(item, worker, vote), ..}*
$b \leftarrow 0$ — *budget spent*
while $b < B$ **do**
    $batch \leftarrow$ select $batchSize$ items from $I (i.e., LI \cup UI)$ via Q
    $VotesBatch \leftarrow$ collect crowd votes for items $batch$
    $VotesAll \leftarrow VotesAll \cup VotesBatch$
    $T \leftarrow AggregateVotes(VotesAll)$ — *build training dataset*
    train M on T
    test M on I
    update UI, LI, b
end while
classify I based on M and $VotesAll$
return M, *Classified Items*

5.2.2 A new approach: *Block Certainty*

Most of the AL approaches do not consider cost-sensitive learning scenarios, in which different types of errors can be associated with different costs. This is the typical scenario in medical screening tests for instance, where we try to uncover the presence of a disease. A *false negative* (FN) error (or missed alarm) is in this case extremely critical and much more costly than a *false positive* (FP) one (false alarm). The same is true in many enterprise contexts such as the ones some of the authors face daily, where companies are ok if a customer request is not understood and has to be routed to an agent, but not ok if ML predicts the wrong intent and gives the wrong answer to the customer. An effective ML classifier for this scenario should be trained using a cost-sensitive loss, that potentially trades *precision* for *recall*. We thus propose a simple cost-sensitive AL approach, that we name "*Block Certainty*", which intrinsically considers the

relative harm of FN over FP errors during the AL process.

Let k indicate the relative harm of FN over FP errors, i.e., a single FN error is k times more costly than a FP one. Let p be the probability of an item being positive according to the machine M. Let d be the decision threshold for M. We first investigate how to set d so as to minimize the risk of the classifier M given k. Assuming $p > d$ (item classified as positive), we define the $risk\ score$ for this item as $risk = 1 - p$. Similarly, the probability of an item being negative is $1 - p$. Assuming $p \leq d$ (item classified as negative), the $risk\ score$ for this item will be $risk = k \cdot p$. Figure 5.1 shows the relation between p and the $risk\ score$. The optimal classification threshold d is given by the value of p for which the lines $risk = k \cdot p$ and $risk = 1 - p$ intersect, i.e., $d = 1/(1 + k)$.

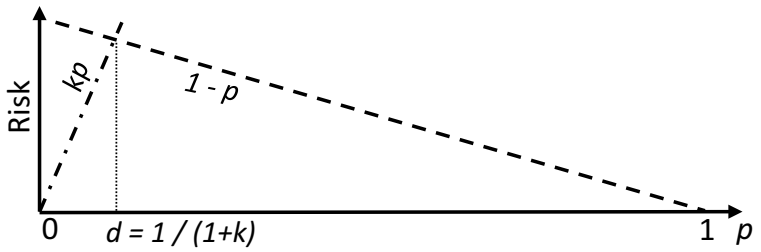


Figure 5.1: $Risk\ score$ Vs. p

Then, in Algorithm 2 we define the "*Block Certainty*" sampling, where at every AL iteration, we query predicted positive and negative items with the lowest $risk\ score$ (according to k) for further annotation.

Algorithm 2 Block Certainty Sampling

Input: I, M, $batchSize$, k

1: p := $M.predictProba(I)$
2: d:= $\frac{1}{1+k}$
3: predPos := [p[i]>d for i in I]
4: predNeg := [p[i]≤d for i in I]
5: PosId := argmax(predPos, size=d·$batchSize$)
6: NegId := argmax(predNeg, size=(1-d)·$batchSize$)
7: **return** PosId + NegId

5.2.3 AL approaches and ML classifier

We examine the following seven AL approaches: (i) *random* (*R*) (items are randomly sampled), (ii) *uncertainty* (*UC*) ([81]), (iii) *certainty* (*C*) (the reverse of uncertainty sampling, i.e., the most certain items are sampled), (iv) *block certainty* (*BC*) (our proposal for cost-sensitive sampling), (v) *QUIRE* (*Q*) ([65]), (vi) *MinExpError* (*MinExp*) ([96]), and (vii) *meta learning* (*LAL*) ([72]). We used *R*, *UC*, *C*, *Q*, and *MinExp* as the SOTA fixed-strategies that have been used in the most of the comparative analysis in the literature. Since [72] already proved that *LAL* approach outperforms the SOTA dynamic strategy approach *ALBL* ([64]), we tested only *LAL* to have an intuition about adaptive approaches.

Since the *LAL* ([72]) approach is specifically designed to work with the random forest classifier, we chose random forests as the underlying classifier for all AL approaches. We used the implementation provided by the *scikit-learn* library ([106]) with the following parameters: n_estimators = 100, criterion = "gini", max_depth = None, bootstrap = True, class_weight = "balanced", and random_state = 2020. In order to evaluate the effect of the choice of the classifier on the performance, we additionally evaluated a subset of the AL approaches (excluding *LAL*) using a support vector machine as the underlying classifier.

5.2.4 Crowdsourcing and Evaluation

Crowdsourcing scenarios.

We consider the following two crowdsourcing scenarios in our experiments:

- *Unlimited votes.* Every single item can be selected an unlimited number of times for a crowdsourced vote, as long as we have available budget and voters.

- *Limited votes.* There is a maximum number of votes $maxVote$ that each single item can receive ($maxVote = 3$ in the experiments). Upon reaching

this limit, the corresponding item is removed from I and cannot be queried anymore. In principle, this limit could be automatically inferred via meta-learning approaches, but this is out of the scope of this thesis.

Evaluation scenarios.

We evaluate the results under two different cases:

- *ML (only)*: We use the trained M to predict the label of each item in the pool I and evaluate its performance compared to the ground truth labels.
- *ML+C*: We take crowdsourced labels for items LI and use M to predict the label of the unlabelled items in UI. We evaluate the performance of this combined crowd-machine classifier by comparing its predictions with the ground truth labels.

Label fusion methods

Since we use crowd answers instead of gold labels in the learning phase, it is important to consider that they may yield low-quality or noisy labels ([160]). In the crowdsourcing literature, this problem is addressed by assigning each task to multiple crowd workers and then aggregating the votes (answers) to obtain the correct label. This process is called truth inference and relies on an aggregation strategy to combine labels. Several label fusion strategies have been investigated in the literature ([12, 46, 30, 50, 83, 87, 90]).

Because of its simplicity and effectiveness, the most popular label fusion method is *Majority Voting* (MV) ([137]), which selects the label voted by the majority of the workers as the correct answer ([53, 104, 91]). We thus use MV as the label fusion strategy in our experimental evaluation. A known limitation of MV is the fact that it assumes that all workers provide the same quality of answers. To measure whether the results depend on this simplifying assumption,

we also ran an additional experimental investigation using a more refined label fusion method (Section 5.2.7).

Metrics.

We use F1, F3, Accuracy, and Loss metrics to evaluate the performance of the machine classifier (*ML*) and of the hybrid crowd-machine classifier (*ML+C*). We define the Loss of classification as the following ([100, 76]):

$$Loss = \frac{1}{|I|} \cdot (k * FNN + FPN), \tag{5.1}$$

where k denotes how much the cost of a false negative outweighs the one of a false positive, FNN is the number of false negatives, FPN is the number of false positives, and $|I|$ is the number of items in I. The Loss summarizes the subjective perspective of the risks for False Positive/Negative errors. This is especially common in real-world applications, such as potential credit card fraud, identifying tweets linked to criminal activities, or literature reviews where screening out a relevant paper is considered to be a serious error affecting the quality of the review, while a falsely included paper just requires some extra work by the authors.

5.2.5 Datasets

Crowdsourced datasets can either include the answer (label) of each crowd worker to each item or just provide the aggregated labels (a single discrete label for each item, determined by combining votes from multiple crowd workers). The latter type of dataset is less informative and doesn't allow to test strategies that involve queries to individual workers. However, most of the available crowdsourced datasets are of this type. We thus created an open repository[2] of the available crowdsourced datasets with individual crowd votes; we also added the datasets we collected. We provide a standard format for accessing

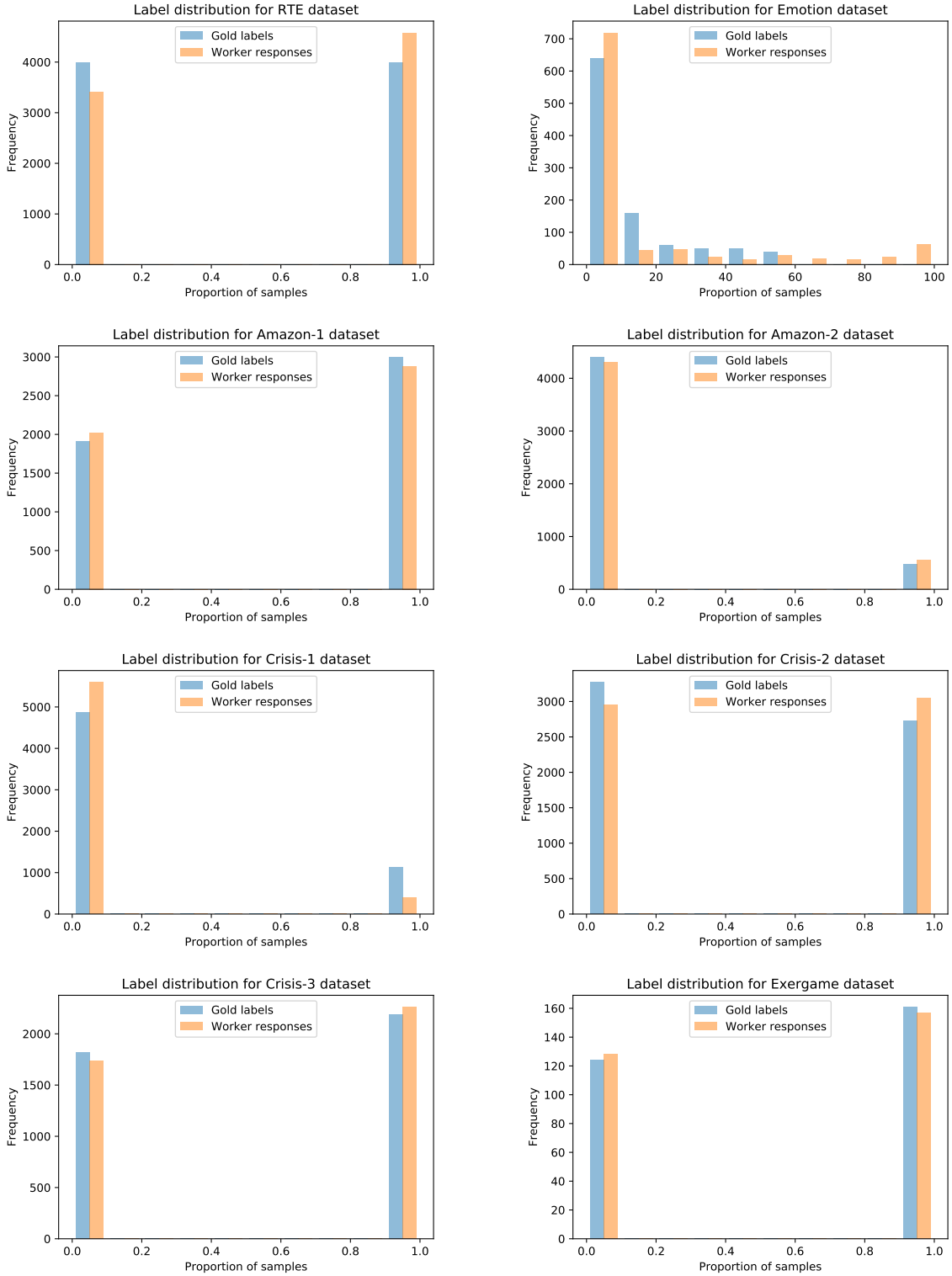


Figure 5.2: Label distributions of datasets

	Datasets							
	RTE	**Emotion**	**Amazon-1**	**Amazon-2**	**Crisis-1**	**Crisis-2**	**Crisis-3**	**Exergame**
Tasks	800	100	1000	998	1948	1948	949	93
Workers	164	38	263	263	79	79	93	38
Total votes	8000	1000	4908	4873	6000	6000	4003	286
Min. vote	10	10	2	2	3	3	3	2
Data (+/-)	400/400	4/96	612/388	99/899	347/1601	883/1065	516/433	53/40
Batch count	40	5	50	50	90	90	50	5
Batch size	20	20	20	20	20	20	20	20
Exp. count	20	20	20	20	20	20	20	20

Table 5.1: Properties of the datasets

the datasets so that they can be used in the experiments without any workload in preprocessing. Researchers can benefit from this repository for hybrid human-machine classification and ranking tasks, truth discovery based on crowdsourced data, estimation of the crowd bias, and active learning.

Table 5.1 shows the properties of each dataset we used in our experiments; the number of tasks, number of workers, total number of votes, minimum vote count per item, data proportion in terms of the number of positives and negatives, batch count (a batch is a predefined number of instances, where the batch count is the number of batches that defines the total number of iterations), size of a batch, and the number of experiments (repetitions) for each dataset. We show the distribution of labels for each dataset in Figure 5.2.

The task in the *Recognizing Textual Entailment* (RTE) dataset ([129]) is to identify whether a given hypothesis sentence is implied by the information in the given text[3].

The *Emotion*[3] dataset ([129]) is about rating the emotion ("anger") of a given text. Each rating is a value between 0 and 100, and we converted them to binary form (0 if $rating \leq 49$, else 1).

The *Amazon Sentiment-1*[4] dataset ([75]) includes annotations about deciding

[3]https://sites.google.com/site/nlpannotations/

[4]https://tinyurl.com/AmazonSentiment

whether the given product review belongs to a book or not. Similarly, the *Amazon Sentiment-2*[4] dataset ([75]) includes annotations about whether the given product review has a negative or positive sentiment.

The *Crisis-1*[5] dataset ([67]) consists of human-labeled tweets collected during the 2012 Hurricane Sandy and the 2011 Joplin tornado. The task is to decide whether the author of the tweet seems to be an eye witness of the event. Similarly, *Crisis-2*[5] dataset ([67]) contains annotations about deciding the type of the message (tweet). The task in *Crisis-3*[5] dataset ([67]) is to analyze hurricane-related tweets and decide whether the tweet is informative or not.

Finally, the *Exergame* dataset includes annotations about whether the given paper describes a study that uses an exergame. An exergame is a form of interactive gaming where people do physical activities while playing a video game, that is, physical exercises by way of video games.

5.2.6 Results

The elaborated experiment results, all datasets, and the source code for reproducing the experiments are available online[1]. In addition, we present the visual experiment results in a notebook[6], while summarizing the important outcomes below.

Table 5.2 and Table 5.3 shows F1 scores of each AL approach in Scenario 1 and Scenario 2, respectively. We observed that *ML+C* prediction outperforms the *ML* prediction on each dataset, regardless of the AL approach. When we compare F1 and F3 scores, the best AL approach remains the same. That is why, we present F1 scores here, while F3 scores can be seen in the results sheet[7].

Comparing the Loss with different k values, we noticed that (i) when the

[5]https://crisisnlp.qcri.org/

[6]https://tinyurl.com/ALExperimentResults

[7]https://tinyurl.com/ALResultSheet

harm of FP and FN is the same ($k = 1$) uncertainty sampling performs 35% better on average than other approaches, (ii) when the problem is characterized by high k value ($k \geq 10$) then block certainty sampling provides 25% better performance on average on five datasets, while uncertainty sampling provides 36% better performance on average across all datasets, and (iii) block certainty sampling approach outperforms others on five datasets with small k values ($k \leq 0.1$) with an improvement of 34.5% with respect to the average performance. We only present the results for $k = 100$ here (see Table 5.4), while keeping others in the notebook[6].

When we analyze the performance of the approaches individually, we draw the following conclusions:

1. Block certainty has an outstanding performance with very big and very small k values ($k \geq 100$ and $k \leq 0.01$) on RTE, Emotion, and Exergame datasets; so it can be used in domains where the harm of a false negative and false positive is very different, such as in literature reviews, or medicine.

2. Certainty and QUIRE ([65]) approaches did not show a promising performance over crowdsourced data in terms of accuracy, F1 and F3 scores.

3. Random sampling performed comparable or sometimes better (i.e. in the Crisis-1 dataset) over imbalanced data with more negatives.

4. Although it is claimed that LAL ([72]) approach outperforms uncertainty sampling ([81]), results show that uncertainty sampling outperforms others in most cases for the finite-pool hybrid classification over crowdsourced data.

5. Table 5.5 shows that uncertainty sampling and random sampling created the biggest number of training sets (note that we may relabel the training

data and we may end up with a different number of votes per item; this affects the size of training data after the learning process ends).

6. The limited votes scenario increased the size of training sets for all cases. This shows that sampling strategies may be stuck at some items if we do not limit the maximum number of votes per item.

		AL approach							
Dataset	**Evaluation**	**R**	**UC**	**C**	**BC (k=1)**	**Q**	**LAL-R**	**LAL-I**	**MinExp**
RTE	ML	66,3	**69,7**	55,4	60	67,2	64	62,9	64,8
	ML+C	68,4	**72,1**	56,7	61,1	67,4	66,7	65,5	66,8
Emotion	ML	32,8	**53**	44,8	51,7	32,7	42,4	44	33,8
	ML+C	42,2	68,4	60,2	**70**	48,5	60,2	56,6	42,7
Amazon-1	ML	93,9	**96,2**	58,6	74,3	70	92,5	92,7	92,4
	ML+C	93,9	**96,3**	59,1	74,5	70,4	92,6	92,8	92,5
Amazon-2	ML	55	63,5	35	36,4	26	**67,6**	65,7	32,1
	ML+C	60,7	70,1	40,2	42,5	31	**72,5**	71,4	34,2
Crisis-1	ML	**31,4**	7,5	14,9	5,9	7,7	27	28,5	11,8
	ML+C	**38,5**	10,3	17,9	9,4	8,2	30,6	32,4	14,2
Crisis-2	ML	80,8	**85,5**	61,8	66,9	15,7	66,1	61,5	79,7
	ML+C	81	**86**	62	67	16,1	66,3	61,6	79,9
Crisis-3	ML	85,2	**89,5**	69,8	68,7	47,6	69,4	73,1	84,1
	ML+C	85,7	**90,2**	69,9	68,7	47,8	69,6	73,3	84,3
Exergame	ML	79,4	**81,5**	74,9	73,4	79,4	77,8	77,4	78,5
	ML+C	80,3	**81,7**	75	73,8	79,4	78,1	77,8	78,7

Table 5.2: F1 Scores in percentage (Scenario 1: Unlimited votes)

5.2.7 Further analysis

The previous experimental analysis was run using random forests as the underlying classifier and majority voting as the label fusion strategy. In this section, we investigate whether changing the classifier or fusion strategy affects the overall picture. Since the LAL ([72]) approach is specifically designed to be used with random forests, we omit it from the following analysis. We also

Dataset	Evaluation	AL approach							
		R	**UC**	**C**	**BC (k=1)**	**Q**	**LAL-R**	**LAL-I**	**MinExp**
RTE	ML	65,4	**70,2**	61,5	64,9	57,6	63,9	63,7	65,6
	ML+C	67,9	**72,5**	63,5	66,7	58,7	66,7	66,4	67,7
Emotion	ML	35,4	44,2	35,8	**44,9**	38,8	41,7	41,1	35,7
	ML+C	44,5	54,6	49,1	**59,8**	51,4	56,9	53,4	47
Amazon-1	ML	93,7	**96,2**	65,8	84,6	80,2	94,3	92,9	94
	ML+C	93,8	**96,3**	66,2	84,8	80,7	94,5	92,9	94,1
Amazon-2	ML	56,6	**70,3**	43,6	63,9	42,8	67,1	66,6	52,9
	ML+C	62,6	**76,7**	50,3	70,7	49,8	72,8	72	59
Crisis-1	ML	**32,2**	22,6	31	21,6	18,8	30,3	28,5	20,9
	ML+C	**39,6**	28,6	36,7	27,1	24,5	35,5	34,5	25,2
Crisis-2	ML	81	**86**	71,1	69,2	73,1	73,7	73,2	80,7
	ML+C	81,1	**86,3**	71,2	69,4	73,3	73,9	73,4	80,8
Crisis-3	ML	85,1	**89,5**	74,5	74,2	68	77,5	77,4	85,2
	ML+C	85,3	**90,1**	74,7	74,6	68,3	77,8	77,5	85,5
Exergame	ML	79,8	**84**	75,3	76,1	80	78,7	78,4	78
	ML+C	79,9	**84,5**	75,4	76,1	80,3	79,5	79,1	78,3

Table 5.3: F1 Scores in percentage (Scenario 2: Limited votes)

Dataset	AL approach							
	R	**UC**	**C**	**BC (k=1)**	**Q**	**LAL-R**	**LAL-I**	**MinExp**
RTE	628,08	427	747,46	**158**	884,11	702,72	712,82	621,86
Emotion	6,24	5,2	5,55	**4,74**	5,3	5,39	5,39	6,14
Amazon-1	274,29	**169,1**	1465,4	1522,11	859,26	260,98	292,06	265,15
Amazon-2	256,16	**176,7**	335,31	347,77	342,4	188,39	189,2	291,47
Crisis-1	**1400,06**	1513	1406,86	1529,07	1553,88	1423,26	1446,3	3062,11
Crisis-2	756,29	**460,36**	1278,99	1454,45	1046,7	1287,1	1302,9	1543,61
Crisis-3	391,57	**240,9**	735,25	597,34	1113,8	685,46	667,1	393,94
Exergame	59,2	45,9	75,97	**42,32**	58,2	66,16	63,23	62,28

Table 5.4: Loss ($k = 100$) (Scenario 2: Limited votes)

	Dataset							
AL approach	**RTE**	**Emotion**	**Amazon 1**	**Amazon 2**	**Crisis 1**	**Crisis 2**	**Crisis 3**	**Exergame**
R	523.6	**70.5**	649.55	649	**1199.9**	1197.9	638.6	72.3
UC	**645.2**	58.45	**806.85**	**742.95**	1129.3	**1478.8**	**768.25**	**84.9**
C	351.65	51.05	404.9	491.6	837.45	772.3	421.65	57.95
BC (k=1)	356.95	50.45	416.75	582.25	976	779.4	423.75	58.7
Q	283.7	52.05	357.85	358.8	622.65	622.65	354.55	61.75
LAL-R	495.65	55.5	629.2	714.25	1191.3	786.05	441	63.8
LAL-I	496.45	59.35	581.5	710.25	1183.6	794.15	434.85	65.15
MinExp	510.2	59	640	474,9	747,4	1181,7	630.15	73.8

Table 5.5: Size of training set (Scenario 2: Limited votes)

		Metric		
Dataset	F	Accuracy	F1	Loss (K=100)
RTE	MV	*RF,UC* **0,69**	*RF,UC* **0,725**	*SVM,R* **134**
	DS	*RF,UC* **0,701**	*RF,UC* **0,701**	*SVM,Q* **184**
Emotion	MV	*SVM,Q* **0,945**	*SVM,UC* **0,599**	*RF,BC* **4,7**
	DS	*RF,BC* **0,96**	*RF,BC* **0,96**	*RF,BC* **4,8**
Amazon-1	MV	*SVM,UC* **0,961**	*SVM,UC* **0,968**	*SVM,UC* **152**
	DS	*RF,UC* **0,954**	*SVM,UC* **0,959**	*RF,UC* **185**
Amazon-2	MV	*SVM,UC* **0,955**	*RF,UC* **0,767**	*RF,UC* **177**
	DS	*RF,UC* **0,956**	*RF,UC* **0,956**	*SVM,UC* **145**
Crisis-1	MV	*RF,R* **0,866**	*RF,R* **0,396**	*RF,R* **1731**
	DS	*SVM,R* **0,859**	*RF,R* **0,858**	*RF,C* **1441**
Crisis-2	MV	*RF,UC* **0,87**	*SVM,UC* **0,865**	*SVM,C* **172,3**
	DS	*SVM,UC* **0,88**	*RF,UC* **0,872**	*SVM,UC* **492**
Crisis-3	MV	*SVM,UC* **0,901**	*SVM,UC* **0,911**	*SVM,UC* **241**
	DS	*SVM,UC* **0,904**	*SVM,UC* **0,914**	*RF,BC* **173**
Exergame	MV	*RF,UC* **0,825**	*RF,UC* **0,845**	*SVM,BC* **33,2**
	DS	*RF,UC* **0,796**	*SVM,UC* **0,812**	*SVM,BC* **35,2**

Table 5.6: Best performance results in ML + C prediction

omit minExpError ([96]) as its complexity is very high and it did not show to be competitive with less expensive approaches. Hence, we focused on the following AL approaches: (i) *random* (R), (ii) *uncertainty* (UC) [81], (iii) *certainty* (C), (iv) *block certainty* (BC), and (v) *QUIRE* (Q) ([65]).

We repeated all experiments using an SVM classifier, utilizing an implementation provided by scikit-learn library ([106]) with the following parameters: class_weight ="balanced" and C=0.1. As an alternative to majority voting we used the *Dawid-Skene* (DS) ([44]) strategy, a popular label fusion method that models each worker as a confusion matrix and uses an expectation-maximization approach to decide the correct label.

Results[8] confirmed that the combination of humans and the machine (*ML + C*) outperforms the machine only case (*ML only*). For this reason, in the following we discuss only the results of the *ML + C* case.

Table 5.6 shows which *(classifier, AL approach)* pair performs best on each dataset in terms of *Accuracy*, *F1*, and *Loss (K=100)* metrics. We evaluate results in two different conditions; when we aggregate votes using (i) majority voting (MV), and (ii) Dawid-Skene (DS). Results show that this pair may change even on the same dataset with respect to the metric used. For example, when we look at the Crisis-1 dataset, random sampling with SVM classifier and DS label fusion method performs best in terms of accuracy while uncertainty sampling with RF classifier is the best in terms of F1 score.

Summing up, these results suggest that the performance that AL strategies exhibit in standard settings cannot be directly transferred to hybrid classification problems. Many factors may affect the behavior of an AL approach, such as the machine classifier being used, the amount of labeling noise in the data, the characteristics of the problem, and the label fusion method. For example, when we look at the F1 scores in Table 5.2 and Table 5.3, we see that uncertainty sampling is not the best method for Emotion, Amazon-2 (in Scenario 1) and

[8]https://tinyurl.com/ComparisonOfAggTech

Crisis-1 datasets. These three datasets are the most unbalanced datasets we used. In addition, the noise level in Emotion and Crisis-1 datasets are very high (please see the Figure 5.2). These observations show that uncertainty sampling performs best when the dataset is balanced and the noise level of the data is low.

5.3 Conclusions and Open Issues

In this chapter, we first reviewed the existing AL approaches under three categories: (i) fixed-strategy approaches, (ii) dynamic-strategy approaches, and (iii) strategy-free approaches. We then investigated the performance of a set of representative approaches for different strategies in the hybrid human-machine classification setting. The aim was to discover if and how the performance of the existing approaches can be transferred into the hybrid classification context.

Experimental results showed that as expected, hybrid human-machine classification always improves over purely machine-based classification. When comparing different AL approaches, however, no clear winner emerges. Even a SOTA meta-learning approach like *LAL* fails to show consistent improvements over the alternatives when evaluated across different datasets. We observed that if we have a finite pool classification problem with noisy crowd labels (i.e. RTE, Emotion, Crisis-1, and Crisis-2 datasets), then we can simply start by picking the RF classifier, UC approach, and DS label fusion method to achieve an acceptable performance (see Table 5.6). In addition, if the cost of FN and FP errors are very different for the problem at hand, then we can consider using a BC approach instead of UC (see Table 5.4, where BC improved the performance on RTE, Emotion, and Exergame datasets with a big k value).

Hybrid crowd-machine classification is promising but needs more investigation. Most of the existing AL approaches assume that enough high-quality labeled data exist. However, gathering high-quality labeled data is challenging, and labels, especially if crowdsourced, can be noisy. For this reason, we an-

alyzed what happens if we use noisy labels instead of gold data in finite-pool hybrid classification problems. We have shown that in the presence of noisy data the conclusions from the SOTA need to be revisited and cannot be taken at face value when data is crowdsourced. This points to the need for further work in the community on what is the role of noise in the success of specific AL algorithms and how much noise they can tolerate before the assumptions and intuitions on which they are based do not hold any longer. The community also needs to analyze how different types of noise (i.e. random noise or biased crowdsourced answers) in data affect the performance of AL, and how the effect of such noise can be smoothed. Indeed, label smoothing can be a promising direction to pursue as it helps to improve the calibration of ML models, which is central for many AL algorithms.

In addition, our analysis highlights several open issues and challenges that we believe can shape future research on this topic, specifically: i) what are the trade-offs between having a smaller but accurately labeled dataset vs a larger but more noisy one? ii) Would a dynamic assessment of crowd accuracy help strategy-free approaches? and, iii) How do we operate at the start of an AL process when we have no idea of the crowd accuracy?

Another direction is that of balancing the AL and human vs machine contribution in hybrid crowd-ML classification problems where the pool of items to classify is finite: here the challenge is deciding when to stop spending our budget for collecting samples to optimize our AL process and obtain a stronger ML model and spend the budget for classifying the remaining items in the pool, by applying the current ML model and resorting to humans for items on which the ML model is unsure about.

In summary, this chapter has pointed to several interesting research paths that we need to undertake if we want to address problems that companies face when developing their ML solutions and that are needed to make AL a mainstream part of ML pipelines.

Chapter 6

Value-aware active learning

6.1 Model Value and Active Learning

Recent papers have outlined the massive practical importance of selective classifier, where machine learning (ML) models have the option of saying "I don't know" and therefore the workflows proceed with a default class [31, 122]. Indeed it is also our experience that, in enterprise workflows, deploying models as selective classifiers is the rule, not the exception. What follows from these simple observations is that accuracy and ROC-related metrics are inadequate to capture the value an ML model brings to a use case, both because we have now an "I don't know option", and because correct, wrong, and "I don't know" answer have different "costs" in practice. Therefore, we need a different way to assess ML.

Model value. In this chapter, we follow the literature in terms of how a model *value* is defined and measured [122]. We assume that the value of a model is given by the following formula:

$$V = \rho V_r + (1 - \rho)(\alpha V_c + (1 - \alpha)V_w) \tag{6.1}$$

where V_r, V_c, and V_w are the value of rejecting an item, classifying it correctly, and classifying it wrongly, respectively. ρ is the proportion of predictions that

are rejected and α is the accuracy.

Reality can be more complex in terms of how the cost is assigned as described in the original paper [122], but this simplified model is both good enough for our purposes and is anyway already at the limit of the level of complexity that business process owners can accept when thinking of the benefit of ML in production.

If we set the value of not having AI (before we put our ML model into production) to 0, and we consider that not having ML is equivalent to having a model that rejects all predictions, then we have $V_r = 0$. We can then define V_w in terms of V_c without loss of generality, as $V_w = -k\dot{V}_c$, where k is a cost ratio of how bad is an error with respect to how good is a correct prediction. With this value formula, ML is helpful if value is positive. With these changes, then the value formula becomes:

$$V = V_c(1-\rho)(\alpha - k(1-\alpha)) \tag{6.2}$$

In the following we assume that rejection of predictions is performed by having the classifier emit a prediction score or confidence, and by having the ML solution workflow filter predictions with confidence lower than a threshold t. This is the case in nearly all approaches we have seen. In this setting, if the model is well calibrated and scores correspond to probability of being correct, then theory shows us that we maximize value by setting $t = k/(k+1)$. If that is not the case, then we can set the threshold by using a validation set, picking the t that maximizes value on that dataset.

Directly from the formulas descend the fact that the greater is k, the more selective we should be in accepting predictions. k is a parameter of the business problem, not of the model, and in general, given an ML model, the value will decrease linearly as k increases if t remains constant. If we adjust t as k increases, then value decreases sub-linearly.

Active learning in the age of value. In this chapter, we study how the

fact that value, and not accuracy, is what we look for in a model and it affects active learning strategies. In particular, we investigate if uncertainty sampling [81] has an edge over random sampling, and if simple alternative strategies can outperform both uncertainty and random sampling. The intuition that drives our work is as follows: the value of a model increases when we increase the proportion of correct classifications *above thresholds*. This means that value is provided by i) pushing correct low-confidence classifications above thresholds, ii) pushing above-threshold wrong classifications to go below t, and iii) turning wrong classification into correct one that are above threshold. This is different than the "traditional" learning goal of turning wrong into right, which is not even present in the three options above. However, the objectives we stated are what matters in practice, and the case where customers set $t = 0$ (which occurs when k=0, an unrealistic cost) are virtually non existent.

What follows from this is that uncertainty sampling may not be the best approach if it tries to focus on low confidence examples, as we may never be able to take examples in that low-confidence region to go above t. In contrast, we may instead decide to focus the attention on examples around t. Furthermore, if k is high, this means that the cost of errors is high. In this case, we may actually attempt to do "certainty sampling", or "above threshold sampling", where the goal is to examine above threshold examples because we cannot afford mistakes there. In all this, an underlying problem is that to increase value we need a model that estimates confidences well, so learning to do that is as important as getting the prediction correct.

Contributions. In this chapter, we make two focused but important points: i) uncertainty sampling performs well for unrealistic cost ratios k very close to zero, but is often worse than random sampling even for fairly small values of k; and ii) *Threshold Oriented Sampling (TOS)* performs well across many datasets when k grows to 1 and above. We show this for a variety of natural language datasets. We also show how the selection of text encoders affects the value of

the resulting model. The companion code is available online[1].

6.2 Related Works

[57] proposed the idea of cost-sensitive AL where the model is charged by a cost for every query and it should learn how to improve the model by compensating the cost of asking labels. This scheme has been improved by assigning different costs per samples either based on the annotation time [134] or based on the number of dialogue turns in dialogue systems [148]. Various approaches have been proposed to minimize the annotation cost, such as using a tree-structured model to jointly estimate the utility and cost of each sample [136], or using a theoretical statistical method to limit the optimal number of samples for a particular class [141]. [93] further proposed an algorithm that minimizes the joint cost of mislabeling and annotating.

The cost-sensitiveness has also been referred as the prediction cost in AL. To minimize the prediction cost, [74] used a regressor trained on the labeled data to predict the cost of possible classes and annotate it with the class that has the smallest estimated prediction cost, and [1] used selective sampling to minimize loss. Researchers also proposed algorithms to define various costs for different types of classification errors and minimize the expected loss [162, 158]. However, none of them evaluated the problem from a *value* perspective to study the impact of the reject option on the quality of AL strategies.

6.3 Experimental Work

6.3.1 Experiment Design

Research questions. We have two goals: i) assess if uncertainty sampling performs as well as indicated in the literature as the cost ratio changes, and ii)

[1] https://tinyurl.com/value-aware-AL

see if and when threshold-oriented sampling outperforms uncertainty sampling. We aim at establishing the above across a range of textual datasets and different text encoders.

Dataset	Task	Classes (Distributions in %)	Training/Validation/Test size
Hate Speech&Offensive Language	Classify tweets as hate-speech/offensive/neither	3 (5.77%, 77.43%,16.8%)	14869/4957/4957
Twitter US Airlines	Detect sentiment of tweets about US Airlines	3 (63%, 21%,16%)	8784/2928/2928
DBPedia Classes	Extract content from Wikipedia	9 (52.3%,8.49%,18.8%,9%,0.7%,7.9%,2.4%,0.15%,0.26%)	15000/5000/5000
Clinc150	Intent classification from text	150 (Balanced)	15000/5000/5000
Coronavirus tweets NLP	Detect sentiment of tweets about coronavirus	5 (13.5%,16%,24.4%,18.6%,27.5%)	26972/8991/8991
News Category	Identify news type based on short descriptions	41 (Unbalanced, 3 dominant classes with 16.7%, 8.5%, 8.1%)	10888/3591/3584

Table 6.1: Statistics of the datasets used in the experiments

AL strategies. Besides using the random sampling (selecting instances randomly from a given unlabeled pool), uncertainty sampling [81], and certainty sampling, we also propose two novel *threshold-oriented* AL strategies.

Uncertainty sampling. In [81] [125, 121], the idea is to sample instances on which the ML model is the most uncertain about. Uncertainty is measured via different techniques, such as entropy [128], margin [123], and the least confidence [81]. We use the margin-based uncertainty sampling which is known to be suitable for multi-class uncertainty sampling [125]: For each item in the unlabeled pool, we measure the margin between the confidence score of the two most likely classes and then select instances with the minimum margin.

Certainty sampling is the opposite of margin-based uncertainty sampling; we pick instances on which the two most likely classes have the max margin.

Threshold-oriented sampling (TOS) is a new method we propose, where the idea is to select instances on which the model confidence on the most likely class is the closest to the theoretical confidence threshold $t = k/(k+1)$.

TOS-Below is a variation of *TOS* and it selects the samples below t that are closest to t - in an attempt to learn how to push samples in that region above the threshold, to positively contribute to value while reducing the problem of sparsity (and thus lack of diversity) in sampling over the threshold for high values of k.

AL setup. We randomly selected one example per class for each dataset before

the AL starts to seed the model. We fix the batch size to 100 and stop AL when all unlabeled samples are selected. We investigate the performance of AL strategies in multi-class classification tasks given $k \in [0, 100]$.

Datasets. Table 6.1 shows the multi-class datasets (with various class distributions) we use; Clinc150 is a balanced dataset while the rest is unbalanced.

ML models. We use Logistic Regression (LogReg) on top of the text encoders because it is simple and can learn from small datasets, a frequent need in AL scenarios. Furthermore, it learns calibrated predictions satisfying our assumption for using the theoretical threshold. In the appendix, we also report results on RoBERTa model.

Text Encoders. We use default Tf-Idf vectorizer from scikit-learn[2] with the ngram range of (1, 3), and the MPNet from Hugging Face[3].

6.3.2 Results

Insights on SOTA AL strategies. We present results in the early batches using TF-IDF as the text encoder. Results with MPNet are qualitatively similar in terms of behavior of AL strategies, and are reported in Appendix A.2.1. Table 6.2 shows results on 4 datasets on which the SOTA strategies have $>= 75\%$ accuracy. On News Category and Corona datasets LogReg perform poorly (regardless of AL strategy), so the differences are small (see appendix A.2.5) but still confirms the results we report here. As we can see, random and certainty sampling outperform uncertainty sampling, and we speculate that this is because uncertainty sampling cannot manage to throw samples above the confidence threshold in most cases. Note that the performance of uncertainty sampling drops significantly even for fairly low value of k, and it catches up other strategies in the later batches, when the number of samples is so high that the AL strategy does not matter much. We obtain the same results if we use

[2]https://tinyurl.com/sklearn-tfidf-vectorizer

[3]huggingface.co/docs/transformers/model_doc/mpnet

MPNet instead of TF-IDF. The additional difference is that MPNet generally performs better, but the conclusions in terms of AL strategy selection do not change across all datasets for all cost settings. We report the full AL pipelines on each dataset with different models and text encoders as figures in Appendix A.2.3.

AL Strategy	Batch	US Airline				Clinc150				DBPedia				Hate Speech			
		Value				Value				Value				Value			
		κ=0	κ=2	κ=4	κ=8	κ=0	κ=2	κ=4	κ=8	κ=0	κ=2	κ=4	κ=8	κ=0	κ=2	κ=4	κ=8
Uncertainty	5	**0.719**	0.14	0.03	0.002	0.173	0.0	0.0	0.0	**0.835**	0.125	0.014	0.002	**0.796**	0.456	0.366	0.064
	10	**0.732**	0.243	0.066	0.01	0.451	0.0	0.0	0.0	**0.898**	0.271	0.058	0.011	**0.869**	0.537	0.249	0.073
	20	**0.752**	0.321	0.149	0.054	**0.665**	0.0	0.0	0.0	**0.934**	0.536	0.221	0.057	0.891	0.632	0.35	0.128
	30	0.761	0.363	0.21	0.095	**0.721**	0.001	0.0	0.0	**0.942**	0.699	0.431	0.182	**0.896**	0.68	0.45	0.191
Random	5	0.694	0.276	0.081	0.003	**0.405**	0.0	0.0	0.0	0.753	0.36	0.147	0.018	0.795	0.51	**0.493**	0.198
	10	0.72	0.304	0.165	0.054	**0.541**	0.0	0.0	0.0	0.834	**0.506**	0.289	0.117	0.83	0.565	0.521	0.332
	20	0.738	0.353	**0.24**	0.119	0.658	0.006	0.0	0.0	0.883	**0.639**	**0.448**	0.246	0.85	0.632	0.551	0.385
	30	0.748	0.367	**0.254**	0.135	0.704	0.023	0.005	0.0	0.905	0.706	**0.53**	0.33	0.865	0.66	0.574	0.408
Certainty	5	0.635	0.064	0.056	0.032	0.052	**0.026**	**0.016**	**0.01**	0.157	0.055	0.042	0.031	0.777	0.35	0.053	0.247
	10	0.631	-0.063	-0.336	0.067	0.075	**0.04**	**0.03**	**0.016**	0.143	0.081	0.082	0.069	0.778	0.351	0.01	0.015
	20	0.632	-0.027	-0.238	-0.016	0.14	**0.055**	**0.036**	**0.022**	0.625	0.142	0.119	0.095	0.777	0.338	-0.042	-0.479
	30	0.635	0.101	0.036	0.126	0.207	**0.073**	**0.047**	0.021	0.436	0.193	0.172	0.144	0.777	0.335	-0.062	-0.734
TOS	5	0.689	0.237	0.14	0.049	0.141	0.004	0.006	0.005	0.794	0.337	0.28	0.023	0.787	0.578	0.454	0.226
	10	**0.732**	0.31	0.193	0.102	0.387	0.011	0.016	0.014	0.869	0.437	0.381	0.197	0.856	0.628	0.549	0.41
	20	0.746	0.346	0.212	**0.15**	0.643	0.034	0.032	0.015	0.93	0.596	0.416	0.376	**0.893**	0.683	0.574	0.484
	30	**0.762**	0.367	0.244	**0.176**	0.669	0.059	0.045	0.02	0.94	**0.72**	0.438	**0.392**	**0.896**	0.708	**0.585**	**0.502**
TOS-Below	5	0.255	**0.277**	**0.159**	**0.077**	0.361	0.002	0.003	**0.011**	0.283	**0.396**	**0.315**	**0.1**	0.388	**0.602**	0.463	**0.375**
	10	0.255	**0.313**	**0.204**	**0.11**	0.361	0.014	0.006	**0.017**	0.283	0.442	**0.393**	**0.302**	0.388	**0.658**	**0.558**	**0.455**
	20	0.255	**0.361**	0.217	0.125	0.361	0.027	0.023	**0.022**	0.283	0.617	0.423	**0.378**	0.388	**0.696**	**0.582**	**0.487**
	30	0.255	**0.377**	**0.254**	0.155	0.361	0.058	0.04	**0.03**	0.283	**0.726**	0.479	0.381	0.388	**0.717**	**0.587**	0.497

Table 6.2: Performance of AL strategies with LogReg + TF-IDF (using theoretical threshold)

TOS quality. Table 6.2 also shows the performance of *TOS* and *TOS-Below* (the corresponding figures can be found in Appendix A.2.4). Results show that *TOS* and *TOS-Below* have a robust behavior across all datasets, showing the best performance for even for a low cost ratio of 2. We speculate that the poor performance of uncertainty sampling may be in part due to using a batch-mode in AL, because uncertainty sampling is biased towards a specific region of the pool and it tends to select correlated samples within a batch [42]. However, performing AL in batches is common in practice.

Tuning threshold on test set. AL quality when model quality is assessed on value rather than accuracy may be due to the quality of model calibration - or lack thereof. To assess this, we experiment AL strategies assuming we ”magi-

cally" pick the threshold that optimizes value, that is, if we could pick it based on the *test* set.

To do this we simply searched the t in [0, 1] that maximizes value. If two different thresholds give the maximum value, we arbitrarily picked the smaller one to maximize the coverage of the selective classifier. The threshold differs only for the DBPedia dataset, where the empirical t is considerably smaller than the theoretical t. On this dataset the model is very accurate and we tend to underestimate confidence. Since we lower the threshold, the performance of uncertainty sampling improves, not because the AL process changes (it doesn't), but because we now do manage to raise the confidence of samples above the threshold. (see Appendix A.2.2). This being said, of course in reality we cannot use a test set for model parameter setting, and in AL scenarios we also likely do not have a large validation set for optimizing thresholds, so we have to do with the theoretical one.

Results with fine-tuned RoBERTa We further investigated what happens if we use AL to fine-tune a large language model rather than keeping the encoding fixed. We used RoBERTa-base model[4] and fine-tuned it for sentiment analysis on the US Airlines dataset. We took the RoBERTa-base model and added one linear layer "$Linear(in\_features = 768, out\_features = 768, bias = True)$" as the pre-classifier, followed by another linear layer "$Linear(in\_features = 768, out\_features = 3, bias = True)$" as the classifier. At every batch, we loaded the network from scratch and trained with the current training set. Results indicate that all AL strategies perform roughly the same in this setting (see Appendix A.2.6). This is in line with the findings of a recent paper [92] that shows that the performance of different AL strategies become very close when applied to the fine-tuning of large language models that have been pre-trained on large enough datasets. They obviously need little learning, or, it is hard to find significant differences with the AL strategies we examine.

[4]https://huggingface.co/roberta-base

6.4 Limitations and Conclusions

In this chapter, we are just scratching the surface of value-oriented AL. We tested a limited number of NLP datasets and a limited set of encoders and algorithms. For this reason, even if TOS seems superior we cannot claim this as a fact with any sort of generality. What we can say however is that we believe we have provided enough evidence to reconsider the superiority of uncertainty sampling, propose TOS as a valid contender, and lay down the motivations for larger scale investigation on both outcomes and reasons behind the outcomes. This is important because i) uncertainty sampling is very widely adopted, but ii) model thresholding is also ubiquitous in practice, and from what we have seen even random sampling is often preferable, and starting from low cost ratio.

Chapter 7

Conclusions and Future Work

In this closing chapter, we summarize our contributions by addressing our solutions to each research question we investigated. Then, we discuss the limitations of our work and present the planned future work.

7.1 Recap of research questions and responses

RQ1. & RQ3. How to measure quality in ML, and is accuracy a good measure of quality in the hybrid classification context? What are the proper metrics for the cost of using ML with rejection?

We discussed that accuracy metrics are optional when ML models are adopted in real-world enterprise scenarios. There is nothing bad in efforts to improve model accuracy: they should stay among the main goals of ML research. However, we argued that one of the reasons for the disconnect between the amazing progress of ML research and the limited adoption of ML in the enterprise is the focus on one aspect of the problem only, and on the fact that we have not paid enough attention to how and why models are used in practice, and to the aspects and metrics that are relevant to enterprises when they adopt and deploy a model. We highlighted that we almost always apply ML models as selective classifiers in enterprise AI workflows, and we need to define reasonable confidence thresholds that depend on the cost of machine errors and their relation to

the cost of rejection and the value of a correct machine prediction. This being the case, we proposed that we are not really focusing on the overall accuracy of the ML models, instead, we care about the accuracy around the confidence threshold. We further proposed the "value" metric to assess the performance of models and showed that a well-calibrated model can have a good value, even if it has arbitrarily bad accuracy.

RQ2. How does the model calibration affect the performance of ML models in hybrid classification contexts? And how can we obtain well-calibrated hybrid classifiers?

We proposed a crowd-powered hybrid classification service that automatically or semi-automatically combines the machine, crowd, and experts in cost-efficient and effective way to solve a finite or infinite pool of classification problems. The classes of tasks we addressed have the option of asking the class of a particular item to humans, though at a cost. Humans can be experts or crowd workers who interact with the system via a crowdsourcing platform (i.e. Amazon Mechanical Turk). In addition, the service has an ML classification component which provides predictions with a negligible cost concerning the cost of asking humans, and the cost of asking humans is significantly less than the cost of making errors.

Having this service at hand, we observed that the prevelant metric for the ML service component is not the accuracy but calibration. In hybrid classification where the cost for one type of error is high (and also high concerning the cost of asking humans), knowing if we can trust the ML service is the key. Furthermore, the ability to estimate what crowd and ML can do and which kind of items they can or cannot classify accurately becomes central to the effectiveness of the service.

We then investigated how the ML service component can be trained to achieve model calibration and reduce the calibration error. We first evaluated the existing label smoothing and soft-target techniques from the calibration point of

view and showed that they improve the model calibration in hybrid classification. We further proposed novel soft-target techniques that take crowd labels as input and output a set of workers' accuracies and a probability assignment (via label fusion techniques) over the classes for every sample. We showed that our proposed soft-target methods improve the model calibration and decrease the expected calibration errors of deep models.

RQ4. How to effectively characterize ML failures?

When we think of ML failures, we can characterize them as "known unknowns" and "unknown unknowns". Known unknowns are the ones that the machine has low confidence in, and we already have a signal that the machine does not know about these particular items. On the other hand, unknown unknowns are the ones that machine makes high confidence predictions but are indeed wrong. We can also refer to this case as high confidence errors, and we know that especially deep learning models generally suffer from this.

With AL techniques, we can solve the known unknowns problem because the model is already aware that it does not have enough confidence in those particular items. Thus, we can use our proposed value-aware AL strategy to address the "known unknowns" problem in the hybrid classification services.

For the problem of "unknown unknowns", we collaborated with Delft University of Technology (TU Delft) and participated in the CATS4ML challenge[1] organized by Google in 2021. During this work, we discovered and submitted the adverse images known as AI blindspots or the unknown unknowns for the machines. We had a part of the Open Images dataset and discovered blindspot examples guided by a list of preselected target labels. Our approach comprises (i) an identification and characterization of image atypicality task that is presented to the crowd worker both as a local view of visually similar images and a global view of images from the class of interest, and ii) an automatic image sampling method that selects a diverse set of atypical images based on vi-

[1] cats4ml.humancomputation.com/

sual features. We showed the effectiveness and cost-efficiency of our approach through controlled crowdsourcing experiments and provided a characterization of image atypicality based on human annotations of 10K images from the Open Images dataset. This work is out of the scope of this thesis, that is why we only provided an abstract of what we did. The full paper is currently under review for CSCW 2022.

RQ5. How to use AL in hybrid classification contexts? Are the existing AL strategies cost-effective, or do we need novel cost-aware AL methods for hybrid classification services?

We reviewed the AL literature and tested the performance of the SOTA AL strategies over crowdsourced datasets within cost-sensitive hybrid classification contexts under two different scenarios: (i) ML only and (ii) hybrid (human+machine). As part of this process, we also created a collection of crowdsourced datasets containing micro-level information (i.e., individual crowd votes) by aggregating the publicly available ones and adding the ones we collected. Results showed two important observations; (i) the SOTA AL strategies do not achieve similar performances across crowdsourced datasets, and (ii) the hybrid classification scenario improves the performance of AL approaches over crowdsourced datasets.

Based on our observations, we designed a novel value(cost)-aware AL strategy *-threshold oriented sampling (TOS)-* which selects items on which the model confidence is closest to the confidence threshold defined by the cost-sensitiveness of the given task. We showed that *TOS* outperforms the state-of-the-art ones when the cost of incorrect predictions substantially outweighs that of abstaining.

RQ6. How to efficiently combine crowdsourcing and machine intelligence?

We explored this research question in a separate research line and investigated how to efficiently combine crowdsourcing and machine intelligence for the problem of document screening, where we need to screen documents with

a set of ML filters. Specifically, we focused on building a set of ML classifiers that review and screen the documents efficiently. We proposed a multi-label AL strategy, named *objective-aware sampling*, for querying unlabelled documents to annotate. Our method makes a decision on which the machine filter needs more training data and how to choose unlabeled items to annotate for minimizing the risk of overall classification errors, rather than a single filter error. We showed that *objective-aware sampling* significantly outperforms the SOTA AL strategies [78].

7.2 Contributions

Our contributions are summarized below:

1. We proposed a design of a hybrid classification service and showed that the system should include (i) an active learning component to decide which items to classify, (ii) a machine classification component to ensure trustworthy selective classification, and (iii) a hybrid intelligence component that efficiently combines human (crowd) and machine intelligence given an objective (e.g. classify as many items as possible given a fixed budget or training the model) [120].

2. We highlighted the importance of calibration in hybrid classification and proposed novel techniques to improve the calibration of human-machine classifiers in cost-sensitive tasks [120].

3. We criticized that current accuracy-based performance metrics do not measure the actual value of hybrid ML models and proposed a novel metric "value" to assess the performance of hybrid classifiers [122].

4. We conducted an extensive experimental study to compare the performance of SOTA ML models and AL strategies on NLP tasks with cost-sensitive scenarios. We showed that the performance of SOTA models

with a high accuracy significantly drops when we use the "value" metric with various cost settings. We further showed that simple models tend to have a better value than complex well-known language models when the cost of errors is high compared to the cost of asking humans.

5. We published a review and experimental analysis of AL strategies over crowdsourced data and showed that hybrid classification improves the overall performance of AL strategies in cost-sensitive hybrid classification problems. We further discussed the open challenges and future research directions and released a library of implementations of the SOTA AL strategies and a collection of benchmarking datasets for hybrid human-machine classification [121].

6. We proposed a value-aware AL strategy to improve the performance in active learning of selective classifiers in cost-sensitive classification tasks.

7.3 Limitations

Our work on building reliable hybrid human-machine classifiers has several limitations.

We first discuss the limitations related to tests for model calibration (Chapter 2); while our experiments and results are promising, they require a deeper investigations. We need to expand experiment to other deep network architectures and get a deeper understanding of what drives the behaviors we are seeing. The same is true for label fusion methods and related datasets and classification problems, including especially the classification problems with a high number of classes and varying degree of noise. Finally, we need to integrate the modules into an hybrid classification service and test it "end to end", progressively building a science that can eventually put classification tasks on autopilot so that crowd and ML, and their integration, becomes a commodity rather than an

art and a hard challenge.

We also have some limitations in our work on rethinking and reevaluating the value of ML models (Chapter 4). We see this work as providing evidence of the problem and outlining the research needs: more studies (especially with large models and in vs out of distribution datasets) are needed to validate the hypothesis and a deeper understanding of how calibration, confidence distribution, and size of validation set affect model value.

Finally, we highlight the limitations in our value-aware AL work (Chapter 6). We have limited insights on fine-tuning pretrained language models in AL setup, and limited set of ML models, text encoders, and AL strategies. We have experimented with simple models, but then with a complex model (i.e. RoBERTa) we may need way more data and AL may be less reasonable in general.

7.4 Future Work

We are planning to continue our work on value-aware AL and explore the performance of pre-trained large language models in cost-sensitive AL tasks.

We further aim at designing a new metric to measure the calibration error of hybrid classifiers. We believe that the existing metrics for model calibration are not applicable in the context of hybrid classification because they rely on accuracy-based metrics.

Finally, we plan to build a smart rejector based on human feedback on machine failures. This can be done in a data-driven way like the usual ML models are trained, or through a hybrid data- and knowledge-driven method that allows for more explicit control over the items on which the prediction should be rejected. The reliability of human feedback should be considered, as to how human-labeled data has been used for ML training.

Bibliography

[1] Alekh Agarwal. Selective sampling algorithms for cost-sensitive multiclass prediction. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1220–1228, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[2] Charu C. Aggarwal, Xiangnan Kong, Quanquan Gu, Jiawei Han, and Philip S. Yu. Chapter 22 active learning: A survey, 2014.

[3] Sweta Agrawal and Amit Awekar. Deep learning for detecting cyberbullying across multiple social media platforms. In Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury, editors, *Advances in Information Retrieval*, pages 141–153, Cham, 2018. Springer International Publishing.

[4] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. Guidelines for human-ai interaction. In *CHI'19*, pages 1–13, 2019.

[5] Atsushi Ando, Satoshi Kobashikawa, Hosana Kamiyama, Ryo Masumura, Yusuke Ijima, and Yushi Aono. Soft-target training with ambiguous emotional utterances for dnn-based speech emotion classification. In *2018 IEEE ICASSP*, pages 4964–4968. IEEE, 2018.

[6] Aymé Arango, Jorge Pérez, and Barbara Poblete. Hate speech detection is not as easy as you may think: A closer look at model validation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 45–54, New York, NY, USA, 2019. Association for Computing Machinery.

[7] Josh M Attenberg, Pagagiotis G Ipeirotis, and Foster Provost. Beat the machine: Challenging workers to find the unknown unknowns. In *AAAI'11*, 2011.

[8] Jean-Yves Audibert and Sébastien Bubeck. Minimax policies for adversarial and stochastic bandits. In *Proceedings of the 22nd Annual Conference on Learning Theory (COLT)*, pages 217–226, January 2009.

[9] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 322–331, 1995.

[10] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2003.

[11] Arkar Min Aung and Jacob Whitehill. Harnessing label uncertainty to improve modeling: An application to student engagement recognition. In *FG*, pages 166–170, 2018.

[12] Bahadir Ismail Aydin, Yavuz Selim Yilmaz, Yaliang Li, Qi Li, Jing Gao, and Murat Demirbas. Crowdsourcing for multiple-choice question answering. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2946–2953, 2014.

[13] Philip Bachman, Alessandro Sordoni, and Adam Trischler. Learning algorithms for active learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 301–310, 06–11 Aug 2017.

[14] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, page 759–760, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[15] Yuval Bahat and Gregory Shakhnarovich. Classification confidence estimation with test-time data-augmentation. *ArXiv*, abs/2006.16705, 2020.

[16] Agathe Balayn, Panagiotis Soilis, Christoph Lofi, Jie Yang, and Alessandro Bozzon. What do you mean? interpreting image classification with crowdsourced concept extraction and analysis. In *TheWebConf'21*, pages 1937–1948, 2021.

[17] Emilio Balda, Arash Behboodi, and Rudolf Mathar. Adversarial examples in deep neural networks: An overview. In *Deep Learning: Algorithms and Applications*, pages 31–65, 01 2020.

[18] Yoram Baram, Ran El-Yaniv, and Kobi Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291, December 2004.

[19] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In *Proceedings of the 13th In-*

ternational Workshop on Semantic Evaluation, pages 54–63, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics.

[20] R. Bendel, S. Higgins, J. Teberg, and David Pyke. Comparison of skewness coefficient, coefficient of variation, and gini coefficient as inequality measures within populations. *Oecologia*, 78:394–400, 03 1989.

[21] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 49–56, 2009.

[22] Alina Beygelzimer, Daniel Hsu, John Langford, and Tong Zhang. Agnostic active learning without constraints. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, pages 199–207, 2010.

[23] Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert Schapire. An optimal high probability algorithm for the contextual bandit problem. *CoRR*, abs/1002.4058, 2010.

[24] Mohamed-Rafik Bouguelia, Yolande Belaïd, and Abdel Belaïd. Identifying and mitigating labelling errors in active learning. In *Pattern Recognition: Applications and Methods*, volume Lecture Notes in Computer Science, page 17. Springer, 2016.

[25] Mohamed-Rafik Bouguelia, Slawomir Nowaczyk, K. C. Santosh, and Antanas Verikas. Agreeing to disagree: active learning with noisy labels without crowdsourcing. *International Journal of Machine Learning and Cybernetics*, 9:1307–1319, 2018.

[26] Anthony Brew, Derek Greene, and Pádraig Cunningham. Using crowdsourcing and active learning to track sentiment in online media. In

Proceedings of the 19th European Conference on Artificial Intelligence, pages 145–150, 2010.

[27] Samuel Budd, Emma C. Robinson, and Bernhard Kainz. A survey on active learning and human-in-the-loop deep learning for medical image analysis. *ArXiv*, abs/1910.02923, 2019.

[28] Michał Bukowski, Jaroslaw Kurek, Izabella Antoniuk, and Albina Jegorowa. Decision confidence assessment in multi-class classification. *Sensors*, 21:3834, 06 2021.

[29] William Callaghan, Joslin Goh, Michael Mohareb, Andrew Lim, and Edith Law. Mechanicalheart: A human-machine framework for the classification of phonocardiograms. *Proc. ACM Hum.-Comput. Interact.*, 2(CSCW), 2018.

[30] Chris Callison-Burch. Fast, cheap, and creative: Evaluating translation quality using amazon's mechanical turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1*, pages 286–295, 2009.

[31] Fabio Casati, Pierre Noel, and Jie Yang. On the value of ml models. In *Neurips workshop on Human Decisions*, 2021.

[32] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[33] Boxing Chen and Colin Cherry. A systematic comparison of smoothing techniques for sentence-level BLEU. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, Maryland, USA, June 2014.

[34] Zhuo Chen, Weisi Lin, Shiqi Wang, Long Xu, and Leida Li. Image quality assessment guided deep neural networks training. *ArXiv*, abs/1708.03880, 2017.

[35] Justin Cheng and Michael S. Bernstein. Flock: Hybrid crowd-machine learning classifiers. In *Proceedings of ACM CSCW*, New York, NY, USA, 2015. ACM.

[36] Jan Chorowski and Navdeep Jaitly. Towards better decoding and language model integration in sequence to sequence models. *CoRR*, abs/1612.02695, 2016.

[37] Hong-Min Chu and Hsuan-Tien Lin. Can active learning experience be transferred? *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 841–846, 2016.

[38] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Mach. Learn.*, 15:201–221, may 1994.

[39] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *J. Artif. Int. Res.*, 4(1):129–145, 1996.

[40] Gabriella Contardo, Ludovic Denoyer, and Thierry Artières. A meta-learning approach to one-step active-learning. In *International Workshop on Automatic Selection, Configuration and Composition of Machine Learning Algorithms*, volume 1998, pages 28–40, Sep 2017.

[41] L.P. Cordella, C. De Stefano, F. Tortorella, and M. Vento. A method for improving classification reliability of multilayer perceptrons. *IEEE Transactions on Neural Networks*, 6(5):1140–1147, 1995.

[42] Sanjoy Dasgupta, Adam Tauman Kalai, and Claire Monteleoni. Analysis of perceptron-based active learning. In Peter Auer and Ron Meir, editors,

Learning Theory, pages 249–263, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[43] Thomas Davidson, Dana Warmsley, M. Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *International AAAI Conference on Web and Social Media (ICWSM)*, 2017.

[44] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C Applied Statistics*, 28(1):20–28, 1979.

[45] C. De Stefano, C. Sansone, and M. Vento. To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(1):84–94, 2000.

[46] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. Zencrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st International Conference on World Wide Web*, pages 469–478, 2012.

[47] S. Deroski, P. Panov, D. Kocev, and L. Todorovski. Probabilistic active learning : Towards combining versatility , optimality and efficiency. In *Proceedings of the 17th International Conference on Discovery Science (DS)*, 2014.

[48] L. Desreumaux and V. Lemaire. Learning active learning at the crossroads? evaluation and discussion. *ArXiv*, abs/2012.09631, 2020.

[49] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Number 57 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, Florida, USA, 1993.

[50] Ju Fan, Guoliang Li, Beng Chin Ooi, Kian-lee Tan, and Jianhua Feng. Icrowd: An adaptive crowdsourcing framework. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1015–1030, 2015.

[51] Meng Fang, Yuan Li, and Trevor Cohn. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, sep 2017.

[52] Meng Fang, Xingquan Zhu, Bin Li, Wei Ding, and Xindong Wu. Self-taught active learning from crowds. In *2012 IEEE 12th International Conference on Data Mining*, pages 858–863, 2012.

[53] Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: Answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pages 61–72, 2011.

[54] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Mach. Learn.*, 28:133–168, September 1997.

[55] Giorgio Fumera and Fabio Roli. Support vector machines with embedded reject option. In *Proceedings of the First International Workshop on Pattern Recognition with Support Vector Machines*, SVM '02, page 68–82, Berlin, Heidelberg, 2002. Springer-Verlag.

[56] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[57] Russell Greiner, Adam J. Grove, and Dan Roth. Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139(2):137–174, 2002.

[58] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1321–1330. JMLR.org, 2017.

[59] Yuhong Guo and Russ Greiner. Optimistic active learning using mutual information. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 823–829, 2007.

[60] David Haussler, Michael Kearns, and Robert Schapire. Bounds on the sample complexity of bayesian learning using information theory and the vc dimension. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, COLT '91, page 61–74, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.

[61] Mark Heitmann, Christian Siebert, Jochen Hartmann, and Christina Schamp. More than a feeling: Benchmarks for sentiment analysis accuracy. *Communication & Computational Methods eJournal*, 2020.

[62] Martin E. Hellman. The nearest neighbor classification rule with a reject option. *IEEE Transactions on Systems Science and Cybernetics*, 6(3):179–185, 1970.

[63] Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. Large-scale text categorization by batch mode active learning. In *Proceedings of the 15th International Conference on World Wide Web*, pages 633–642, 2006.

[64] Wei-Ning Hsu and Hsuan-Tien Lin. Active learning by learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2659–2665, 2015.

[65] Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. Active learning by querying informative and representative examples. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, pages 892–900, 2010.

[66] Nguyen Quoc Viet Hung, Nguyen Thanh Tam, Lam Ngoc Tran, and Karl Aberer. An evaluation of aggregation techniques in crowdsourcing. In *International Conference on Web Information Systems Engineering*, volume 8181, pages 1–15, 10 2013.

[67] Muhammad Imran, Shady Elbassuoni, Carlos Castillo, Fernando Diaz, and Patrick Meier. Practical extraction of disaster-relevant information from social media. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1021–1024, 2013.

[68] Heinrich Jiang, Been Kim, Melody Y. Guan, and Maya Gupta. To trust or not to trust a classifier. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 5546–5557, Red Hook, NY, USA, 2018. Curran Associates Inc.

[69] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 09 2021.

[70] Mark Johnson, Peter Anderson, Mark Dras, and Mark Steedman. Predicting accuracy on large datasets from smaller pilot data. In *ACL*, pages 450–455, 2018.

[71] Ece Kamar, Severin Hacker, and Eric Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. AAMAS '12, pages 467–474, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[72] Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. Learning active learning from data. In *Advances in Neural Information Processing Systems 30*, pages 4225–4235, 2017.

[73] Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. Discovering general-purpose active learning strategies. *CoRR*, abs/1810.04114, 2018.

[74] Akshay Krishnamurthy, Alekh Agarwal, Tzu-Kuo Huang, Hal Daumé, and John Langford. Active learning for cost-sensitive classification. *Journal of Machine Learning Research*, 20(1):2334–2383, jan 2019.

[75] Evgeny Krivosheev, Fabio Casati, Marcos Baez, and Boualem Benatallah. Combining crowd and machines for multi-predicate item screening. *Proc. ACM Hum.-Comput. Interact.*, 2, nov 2018.

[76] Evgeny Krivosheev, Fabio Casati, and Boualem Benatallah. Crowd-based multi-predicate screening of papers in literature reviews. In *Proceedings of the 2018 World Wide Web Conference*, pages 55–64, 2018.

[77] Evgeny Krivosheev, Fabio Casati, and Alessandro Bozzon. Active hybrid classification. *ArXiv*, abs/2101.08854, 2021.

[78] Evgeny Krivosheev, Burcu Sayin, Alessandro Bozzon, and Zoltan Szlavik. Active learning from crowd in document screening. In *Crowd Science Workshop at 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

[79] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Eric Horvitz. Identifying unknown unknowns in the open world: representations and policies for guided exploration. In *AAAI'17*, pages 2124–2132, 2017.

[80] Walter S. Lasecki, Christopher D. Miller, Iftekhar Naim, Raja Kushalnagar, Adam Sadilek, Daniel Gildea, and Jeffrey P. Bigham. Scribe: Deep

integration of human and machine intelligence to caption speech in real time. *ACM Communications*, 60(9), 2017.

[81] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

[82] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 661–670, New York, NY, USA, 2010. Association for Computing Machinery.

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

[84] Q Vera Liao, Daniel Gruen, and Sarah Miller. Questioning the ai: informing design practices for explainable ai user experiences. In *CHI'20*, pages 1–15, 2020.

[85] Anthony Liu, Santiago Guerra, Isaac Fung, Gabriel Matute, Ece Kamar, and Walter Lasecki. Towards hybrid human-ai workflows for unknown unknown detection. In *TheWebConf'20*, pages 2432–2442, 2020.

[86] Ming Liu, Wray Buntine, and Gholamreza Haffari. Learning how to actively learn: A deep imitation learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1874–1883, jul 2018.

[87] Qiang Liu, Jian Peng, and Alexander Ihler. Variational inference for crowdsourcing. In *Proceedings of the 25th International Conference*

on Neural Information Processing Systems - Volume 1, pages 692–700, 2012.

[88] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

[89] Michal Lukasik, Srinadh Bhojanapalli, Aditya Krishna Menon, and Sanjiv Kumar. Does label smoothing mitigate label noise? In *International Conference on Machine Learning*, 2020.

[90] Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 745–754, 2015.

[91] Adam Marcus, Eugene Wu, Samuel Madden, and Rob Miller. Crowdsourced databases: Query processing with people. In *CIDR*, pages 211–214, 2011.

[92] Katerina Margatina, Loic Barrault, and Nikolaos Aletras. On the importance of effectively adapting pretrained language models for active learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 825–836, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[93] Dragos D. Margineantu. Active cost-sensitive learning. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, page 1622–1623, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[94] Andrew McCallum and Kamal Nigam. Employing em and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 350–358, 1998.

[95] Mauro Mezzini. Empirical study on label smoothing in neural networks. 2018.

[96] Barzan Mozafari, Purnamrita Sarkar, Michael J. Franklin, Michael I. Jordan, and Samuel Madden. Scaling up crowd-sourcing to very large datasets: A case for active learning. *Proc. VLDB Endow.*, 8:125–136, 2014.

[97] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? In *NeurIPS*, 2019.

[98] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2901–2907. AAAI Press, 2015.

[99] Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 78, New York, NY, USA, 2004. Association for Computing Machinery.

[100] A. T. Nguyen, Byron C. Wallace, and Matthew Lease. Combining crowd and expert labels using decision theoretic active learning. In *Proceedings of the Third AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 2015.

[101] Jeremy Nixon, Michael W. Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. *ArXiv*, abs/1904.01685, 2019.

[102] Kunkun Pang, Mingzhi Dong, Yang Wu, and Timothy Hospedales. Dynamic ensemble active learning: A non-stationary bandit with expert advice. In *ICPR*, pages 2269–2276, 2018.

[103] Kunkun Pang, Mingzhi Dong, Yang Wu, and Timothy M. Hospedales. Meta-learning transferable active learning policies by deep reinforcement learning. *CoRR*, abs/1806.04798, 2018.

[104] Aditya Ganesh Parameswaran, Hyunjung Park, Hector Garcia-Molina, Neoklis Polyzotis, and Jennifer Widom. Deco: Declarative crowdsourcing. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1203–1212, dec 2012.

[105] Charles Parker. An analysis of performance measures for binary classifiers. *2011 IEEE 11th International Conference on Data Mining*, pages 517–526, 2011.

[106] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[107] Joshua C Peterson, Ruairidh M Battleday, Thomas L Griffiths, and Olga Russakovsky. Human uncertainty makes classification more robust. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9617–9626, 2019.

[108] Maithra Raghu, Katy Blumer, Greg Corrado, Jon M. Kleinberg, Ziad Obermeyer, and Sendhil Mullainathan. The algorithmic automation problem: Prediction, triage, and human effort. *CoRR*, abs/1903.12220, 2019.

[109] Jorge Ramirez, Evgeny Krivosheev, Marcos Baez, Fabio Casati, and Boualem Benatallah. Crowdrev: a platform for crowd-based screening of literature reviews. *arXiv preprint arXiv:1805.12376*, 2018.

[110] Jorge Ramírez, Burcu Sayin, Marcos Baez, Fabio Casati, Luca Cernuzzi, Boualem Benatallah, and Gianluca Demartini. On the state of reporting in crowdsourcing experiments and a checklist to aid current practices. *Proceedings of The ACM on Human-Computer Interaction (CSCW 2021)*, 5(CSCW2), oct 2021.

[111] Sachin Ravi and Hugo Larochelle. Meta-learning for batch mode active learning. In *6th International Conference on Learning Representations, ICLR 2018, Workshop Track Proceedings*, 2018.

[112] Filipe Rodrigues, Mariana Lourenco, Bernardete Ribeiro, and Francisco C Pereira. Learning supervised topic models for classification and regression from crowds. *IEEE transactions on PAMI*, 39(12), 2017.

[113] Filipe Rodrigues and Francisco C Pereira. Deep learning from crowds. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[114] Carlos Rodriguez, Florian Daniel, and Fabio Casati. Crowd-based mining of reusable process model patterns. In Shazia Sadiq, Pnina Soffer, and Hagen Völzer, editors, *Business Process Management*, pages 51–66. Springer International Publishing, 2014.

[115] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *ICML*, pages 894–905, 2001.

[116] Sebastian Ruder and Barbara Plank. Strong baselines for neural semi-supervised learning under domain shift. In *The 56th Annual Meeting of*

the Association for Computational Linguistics (ACL 2018), pages 1044–1054, 01 2018.

[117] Ognjen Rudovic, Meiru Zhang, Björn W. Schuller, and Rosalind W. Picard. Multi-modal active learning from human data: A deep reinforcement learning approach. *CoRR*, abs/1906.03098, 2019.

[118] Maytal Saar-Tsechansky and Foster Provost. Active sampling for class probability estimation and ranking. *Machine Learning*, 54:153–178, 2004.

[119] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[120] Burcu Sayin, Evgeny Krivosheev, Jorge Ramírez, Fabio Casati, Ekaterina Taran, Veronika Malanina, and Jie Yang. Crowd-powered hybrid classification services: Calibration is all you need. In *2021 IEEE International Conference on Web Services (ICWS)*, pages 42–50, 2021.

[121] Burcu Sayin, Evgeny Krivosheev, Jie Yang, Andrea Passerini, and Fabio Casati. A review and experimental analysis of active learning over crowdsourced data. *Journal of Artificial Intelligence Review*, 54:5283–5305, 2021.

[122] Burcu Sayin, Jie Yang, Andrea Passerini, and Fabio Casati. The science of rejection: A research area for human computation. In *The 9th AAAI Conference on Human Computation and Crowdsourcing*, HCOMP 2021. AAAI Press, 2021.

[123] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *Proceedings of the 4th*

International Conference on Advances in Intelligent Data Analysis, IDA '01, page 309–318, Berlin, Heidelberg, 2001. Springer-Verlag.

[124] Andrew I. Schein and Lyle H. Ungar. Active learning for logistic regression: An evaluation. *Machine Learning*, 68:235–265, 2007.

[125] Burr Settles. Active learning literature survey. In *University of Wisconsin, Madison*, volume 52, 07 2010.

[126] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, 2008.

[127] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 287–294, 1992.

[128] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[129] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, oct 2008.

[130] Lisheng Sun-Hosoya, Isabelle Guyon, and Michèle Sebag. Activmetal: Algorithm recommendation with active meta learning. In *IAL 2018 workshop, ECML PKDD*, 2018. Poster.

[131] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,*, 2016.

[132] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv*, abs/1312.6199, 2013.

[133] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. *ArXiv*, abs/1709.01686, 2017.

[134] Katrin Tomanek and Udo Hahn. A comparison of models for cost-sensitive active learning. In *Coling 2010: Posters*, pages 1247–1255, Beijing, China, August 2010. Coling 2010 Organizing Committee.

[135] M. Tsai, C. Ho, and C. Lin. Active learning strategies using svms. *Wiley Int. Rev. Data Min. and Knowl. Disc.*, pages 313–326, July 2010.

[136] Yu-Lin Tsou and Hsuan-Tien Lin. Annotation cost-sensitive active learning by tree sampling. *Mach. Learn.*, 108(5):785–807, may 2019.

[137] Jinzheng Tu, Guoxian Yu, C. Domeniconi, J. Wang, Guoqiang Xiao, and M. Guo. Multi-label crowd consensus via joint matrix factorization. *Knowledge and Information Systems*, 62:1341–1369, 2019.

[138] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NIPS*, 06 2017.

[139] Thuy-Trang Vu, Ming Liu, Dinh Phung, and Gholamreza Haffari. Learning how to active learn by dreaming. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4091–4101, jul 2019.

[140] Liwei Wang. Smoothness, disagreement coefficient, and the label complexity of agnostic active learning. *J. Mach. Learn. Res.*, 12(null):2269–2292, July 2011.

[141] Min Wang, Yao Lin, Fan Min, and Dun Liu. Cost-sensitive active learning through statistical methods. *Inf. Sci.*, 501(C):460–482, oct 2019.

[142] Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California, June 2016. Association for Computational Linguistics.

[143] Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. The multidimensional wisdom of crowds. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, NIPS'10, page 2424–2432, Red Hook, NY, USA, 2010. Curran Associates Inc.

[144] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. 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.

[145] Bryan Wilder, Eric Horvitz, and Ece Kamar. Learning to complement humans. In *IJCAI'20*, pages 1526–1533, 2020.

[146] Mark Woodward and Chelsea Finn. Active one-shot learning. In *NIPS 2016, Deep Reinforcement Learning Workshop*, 2017.

[147] Ellery Wulczyn, Nithum Thain, and Lucas Dixon. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399, 2017.

[148] Kaige Xie, Cheng Chang, Liliang Ren, Lu Chen, and Kai Yu. Cost-sensitive active learning for dialogue state tracking. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 209–

213, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[149] Yanbing Xue and Milos Hauskrecht. Efficient learning of classification models from soft-label information by binning and ranking. In *The Thirtieth International Flairs Conference*, 2017.

[150] Songbai Yan, K. Chaudhuri, and T. Javidi. The label complexity of active learning from observational data. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019.

[151] Songbai Yan, Kamalika Chaudhuri, and Tara Javidi. Active learning from imperfect labelers. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2136–2144, 2016.

[152] Yan Yan, Rómer Rosales, Glenn Fung, Mark Schmidt, Gerardo Hermosillo, Luca Bogoni, Linda Moy, and Jennifer Dy. Modeling annotator expertise: Learning when everybody knows a bit of something. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 932–939, 2010.

[153] Jie Yang, Thomas Drake, Andreas Damianou, and Yoelle Maarek. Leveraging crowdsourcing data for deep active learning an application: Learning intents in alexa. In *Proceedings of the 2018 World Wide Web Conference*, pages 23–32, 2018.

[154] Biqiao Zhang, Georg Essl, and Emily Mower Provost. Predicting the distribution of emotion perception: capturing inter-rater variability. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 51–59, 2017.

[155] He Zhang, François Petitjean, and Wray Buntine. Hierarchical gradient smoothing for probability estimation trees. In *Advances in Knowledge*

Discovery and Data Mining, pages 222–234, Cham, 2020. Springer International Publishing.

[156] He Zhang, François Petitjean, and Wray Buntine. Bayesian network classifiers using ensembles and smoothing. *Knowledge and Information Systems*, 03 2020.

[157] Mike Zhang, Roy David, Leon Graumans, and Gerben Timmerman. Grunn2019 at SemEval-2019 task 5: Shared task on multilingual detection of hate. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 391–395, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics.

[158] Yexun Zhang, Yanfeng Wang, Wenbin Cai, Siyuan Zhou, and Ya Zhang. From theory to practice: Efficient active cost-sensitive classification with expected error reduction. In *Proceedings of the 2017 SIAM International Conference on Data Mining (SDM)*, pages 153–161, 2017.

[159] Liyue Zhao, Gita Reese Sukthankar, and Rahul Sukthankar. Incremental relabeling for active learning with noisy crowdsourced annotations. *2011 IEEE Third Int'l Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Int'l Conference on Social Computing*, pages 728–733, 2011.

[160] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: Is the problem solved? *Proc. VLDB Endow.*, 10(5):541–552, January 2017.

[161] Jinhong Zhong, Ke Tang, and Zhi-Hua Zhou. Active learning from crowds with unsure option. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1061–1067, 2015.

[162] Siyuan Zhou and Ya Zhang. Active learning for cost-sensitive classification using logistic regression model. In *2016 IEEE International Conference on Big Data Analysis (ICBDA)*, pages 1–4, 2016.

[163] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.

Appendix A

Appendices

A.1 Supplemental material for Chapter 4

A.1.1 Datasets and the experiment flow

Algorithm 3 shows the flow of our experiments. Table A.1 shows the details of each dataset used in our experiments.

Algorithm 3 Experiment flow

for each $(model, task)$ pair **do**
    Train or fine-tune the model using the training set
    Analyze the model's confidence distribution on the test set
    Analyze the model's performance on the test set based on traditional metrics, such as accuracy, precision, ECE, etc.
    Perform a value-based analysis considering different values of k:
    for each k **do**
        Find the theoretical confidence threshold t based on k ($t = k/k + 1$)
        Find the empirical confidence threshold $t_{empirical}$ on the validation set (find the threshold that maximizes the value)
        Find the **value** (based on t) and the **empirical value** (based on $t_{empirical}$) on the test set
    end for
    Output the Value function (plot Value vs k)
    Plot confidence distributions
end for

Dataset	Task	Classes	Distribution	Train/Val/Test size
Hate-speech detection	Classifying tweets as "hate", and "non-hate"	2	Unbalanced	(7734/5068)/(967/1000)/(967/3000)
Clickbait detection	Classifying Twitter posts to detect clickbait	2	Unbalanced	17600/4395/18979
Multi-Domain Sentiment Analysis	Sentiment analysis on Amazon product reviews (DVD, Books, Electronics, Kitchen)	2	Balanced	2000/200/(3386/4265/5481/5745)
DBPedia Classes[1]	Extract structured content from the information created in Wikipedia	9	Highly unbalanced	240942/36003/60794
Yelp-5[2]	Classify yelp text reviews to 5-star ratings	5	Balanced	600000/50000/50000
Hate Speech&Offensive Language[3]	Classifying tweets as hate-speech, offensive language, or neither	3	Unbalanced	14869/4957/4957
Twitter US Airline Sentiment[4]	Detect the sentiment of tweets about major US Airlines	3	Unbalanced	8784/2928/2928
Coronavirus tweets NLP[5]	Detect sentiment of tweets about coronavirus	5	Slightly balanced	26972/8991/8991
News Category Dataset[6]	Identify the type of news based on short descriptions	41	Unbalanced	10888/3591/3584
Clinc150[7]	Intent classification from text	150	Balanced	15000/5000/5000

Table A.1: Statistics of the datasets used in the experiments

A.1.2 Models

For each task, we tested various models as explained below:

- For the hate-speech dataset, we test the following SOTA models: (i) Badjatiya et al. [14] which uses a Recurrent Neural Network (an Embedding Layer (dimension=200) followed by an LSTM network (dimension=50) and a fully connected layer with 3 neurons plus a Softmax) to construct word embeddings and then classify them with Gradient-Boosted Decision Tree. In the original paper, test accuracy is measured as the average of the ten folds in cross validation; however, in our reproduction we separated validation and test set before cross validation, and they are used for evaluation only after training. (ii) one model from Agrawal and Awekar [3] which is composed of an Embedding Layer (dimension=50), followed by a Bidirectional LSTM network (dimension=50), and a fully connected layer of 3 neurons with a Softmax activation. Both models use Dropout (probabilities 0.25 and 0.5, respectively) for regularization, cross-entropy as the loss function, and the Adam optimizer (10 epochs).

[1] www.kaggle.com/datasets/danofer/dbpedia-classes
[2] huggingface.co/datasets/yelp_review_full
[3] https://tinyurl.com/hateSpeech-and-Offensive
[4] https://tinyurl.com/twitterAirlineSentimentDataset
[5] https://tinyurl.com/covid19NLPData
[6] www.kaggle.com/datasets/rmisra/news-category-dataset
[7] archive.ics.uci.edu/ml/datasets/CLINC150

- For the clickbait detection dataset, we test 4 models from one leaderboard team on clickbait challenge: "fullnetconc", "weNet", "lingNet", and "full-Net" which are published on Github[8]. This team modified the task into binary classification - they categorized items with a score under 0.5 into "non-clickbaiting", vice versa.

- For the MDS dataset, we referred to the leaderboard for the sentiment analysis task of "Domain adaptation"[9] and tested the best-performing leaderboard model "*Multi-task tri-training (mttri)*" by Ruder et. al. [116] that is an MLP model with one hidden layer of 50 dimensions, sigmoid activations, and a softmax output. There were 3 other models in the list but their source codes were not published. We further tested two transformer models: (i) Google's T5-base model[10] (12-layers, 768-hidden-state, 3072 feed-forward hidden-state, 12-heads, 220M parameters) fine-tuned on IMDB dataset[11] for sentiment analysis task, and (ii) SieBERT[12] [61]: a fine-tuned version of RoBERTa-large[13] model [88] (24-layer, 1024-hidden-state, 16-heads, 355M parameters) for sentiment analysis task that is fine-tuned and evaluated on 15 diverse text sources. Finally, we tested the simple Logistic Regression (LogR) and 2 different MLP models from scikit-learn library[14] (all parameters we used are explained below).

- For the 7 multi-class datasets, we tested the LogR and 2 MLP models from scikit-learn with the following parameters:

  - **LogR**: We use the default parameters except the max_iter. We increased the max_iter from 100 to 1000 as it did not converge on some

[8] github.com/clickbait-challenge/blobfish
[9] nlpprogress.com/english/domain_adaptation.html
[10] https://tinyurl.com/t5-base-finetuned-sentiment
[11] huggingface.co/datasets/imdb
[12] https://tinyurl.com/SieBERT-sentiment
[13] https://huggingface.co/roberta-large
[14] https://scikit-learn.org/

datasets with 100 iterations.

- **MLP1**: We use the default parameters except one; we set $early\_stopping = True$ in our experiments.
- **MLP4**: The only difference from MLP1 is the modification on the number of hidden layers; we set $hidden\_layer\_sizes = (100, 100, 100, 100)$. This parameter is set to $hidden\_layer\_sizes = (768, 384, 192, 192)$ in MDS experiments.

A.1.3 Experiment details on Hate Speech Dataset

We conducted two different experiments on Hate Speech dataset:

- *Experiment 1*: We used [142] dataset for training, validation, and test set, but we could only recover 9671 of the tweets as of October 2021 (the dataset size is 14949 in the original paper).
- *Experiment 2*: We further analyzed their performance in *Experiment 2* based on the observations of Arango et al. [6]: we used their new dataset (5068 retrieved) as training set. For validation and test set, used the SemEval 2019 dataset from the "Multilingual detection of hate speech against immigrants and women in Twitter" task [19].

A.1.4 Text encoders

We used various text encoders:

- *TF-IDF*: The TfIdf vectorizer of sklearn[15] with the following parameters; "min_df=0, max_features = 1024, strip_accents='unicode', analyzer='word', token_pattern=$r'w\{1, \}'$, ngram_range=(1, 1), use_idf=1, smooth_idf=1, sublinear_tf=1, stop_words='english', lowercase=False",

[15]https://tinyurl.com/sklearn-tfidf-vectorizer

- *MPNET*: The transformer model from Hugging Face[16] with default parameters,
- *nnlm*: Google's universal sentence embedding model that is trained on English Google News 200B corpus, accessible via TensorFlow[17].

A.1.5 GPT-3 Experiments

Since GPT-3 is producing human-like text given an input, we fine-tuned it using the OpenAI API[18]. First, we prepared the MDS dataset for GPT-3; we cleaned sentences that have more than 2049 tokens, and renamed the text column as "*prompt*" and the ground truth column as "*completion*". Then, we used OpenAI API to fine-tune GPT-3 seperately on each of the 4 domains (DVD, books, electronics, and kitchen). We specified "$classification\_n\_classes$" parameter as 2 and $classification\_positive\_class$ as ' 1', so that the API tunes GPT-3 for a binary sentiment analysis. Fine-tuning 4 models on MDS dataset costs a total of $7.15.

[16] huggingface.co/docs/transformers/model_doc/mpnet
[17] https://tinyurl.com/google-nnlm
[18] https://openai.com/api/

In order to test the fine-tuned models on different target domains, we specified the *prompt* in the format of "sentence + -> " because the API itself uses " ->" sign to teach GPT-3 that the sentiment for a *prompt* is (' ->') the *completion*. Thus, fine-tuned GPT-3 models produces either 0 or 1 for the given input. Testing each fine-tuned model on the other 3 domains (so, 12 cases in total) costs $43.89. We provide our source code on Github[19] to show every step of using GPT-3 in our experiments. Figure A.3 shows the average values of GPT-3 on each domain.

A.1.6 Further Results

Binary datasets - Supplementary Results

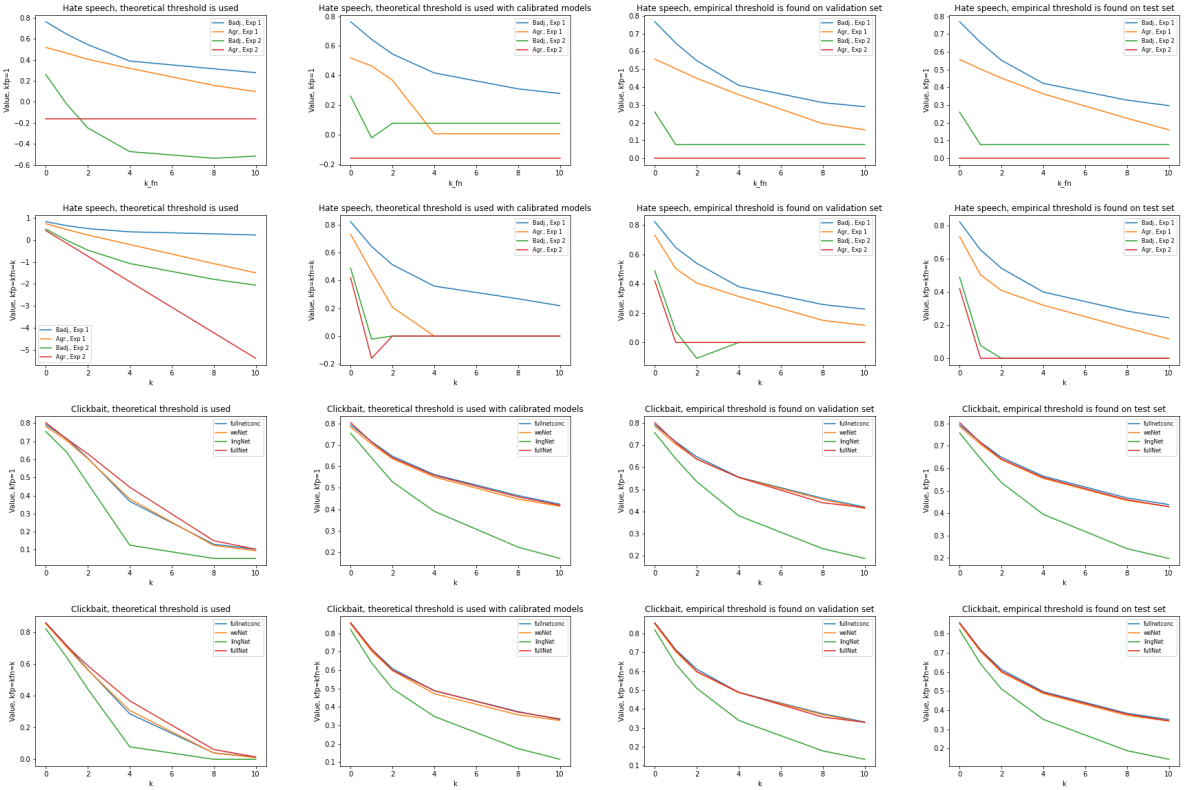


Figure A.1: Value curves of binary datasets for increasing k.

[19] https://tinyurl.com/rethinking-value-of-ml-models

TASK	MODEL	ACCURACY	VALUE$_1$	VALUE$_2$	VALUE$_4$	VALUE$_8$	VALUE$_{10}$
		($k_{fn} = k_{fp} = 0$)	($k_{fn} = 1$)	($k_{fn} = 2$)	($k_{fn} = 4$)	($k_{fn} = 8$)	($k_{fn} = 10$)
HATE-SPEECH	BADJ. ET AL., *Exp. 1*	**0.822**	**0.644**	**0.545**	**0.389**	**0.315**	**0.278**
	AGR. ET AL., *Exp. 1*	0.732	0.464	0.405	0.32	0.157	0.098
	BADJ. ET AL., *Exp. 2*	**0.489**	**-0.022**	-0.248	-0.473	-0.537	-0.516
	AGR. ET AL., *Exp. 2*	0.42	-0.16	**-0.16**	**-0.16**	**-0.16**	**-0.16**
CLICKBAIT	FULLNETCONC	**0.857**	**0.715**	0.608	0.368	0.131	**0.103**
	WENET	0.852	0.703	0.604	0.381	0.124	0.094
	LINGNET	0.82	0.64	0.467	0.125	0.052	0.052
	FULLNET	0.856	0.713	**0.631**	**0.446**	**0.15**	**0.103**

Table A.2: Performance of SOTA models, with theoretical threshold (k_{fp} is set to 1, and $k_{fn} \in \{1, 2, 4, 8, 10\}$).

TASK	MODEL	ACCURACY	VALUE$_1$	VALUE$_2$	VALUE$_4$	VALUE$_8$	VALUE$_{10}$
		($k_{fn} = k_{fp} = 0$)	($k_{fn} = 1$)	($k_{fn} = 2$)	($k_{fn} = 4$)	($k_{fn} = 8$)	($k_{fn} = 10$)
HATE-SPEECH	BADJ. ET AL., *Exp. 1*	**0.822**	**0.644**	**0.51**	**0.362**	**0.272**	**0.217**
	AGR. ET AL., *Exp. 1*	0.732	0.464	0.22	-0.213	-1.081	-1.499
	BADJ. ET AL., *Exp. 2*	**0.489**	**-0.022**	**-0.469**	**-1.077**	**-1.793**	**-2.06**
	AGR. ET AL., *Exp. 2*	0.42	-0.16	-0.74	-1.901	-4.221	-5.382
CLICKBAIT	FULLNETCONC	**0.857**	**0.715**	0.564	0.286	0.041	0.013
	WENET	0.852	0.703	0.561	0.306	0.04	0.011
	LINGNET	0.82	0.64	0.442	0.079	0.0	0.0
	FULLNET	0.856	0.713	**0.588**	**0.367**	**0.061**	**0.015**

Table A.3: Performance of SOTA models, with theoretical threshold (k_{fp} = $k_{fn} \in \{1, 2, 4, 8, 10\}$).

TASK	MODEL	ACCURACY	VALUE$_1$	VALUE$_2$	VALUE$_4$	VALUE$_8$	VALUE$_{10}$
		($k_{fn} = k_{fp} = 0$)	($k_{fn} = 1$)	($k_{fn} = 2$)	($k_{fn} = 4$)	($k_{fn} = 8$)	($k_{fn} = 10$)
HATE-SPEECH	BADJ. ET AL., *Exp. 1*	**0.822**	**0.644**	**0.545**	**0.417**	**0.309**	**0.278**
	AGR. ET AL., *Exp. 1*	0.732	0.464	0.367	0.005	0.005	0.005
	BADJ. ET AL., *Exp. 2*	**0.489**	**-0.022**	**0.077**	**0.077**	**0.077**	**0.077**
	AGR. ET AL., *Exp. 2*	0.42	-0.16	-0.16	-0.16	-0.16	-0.16
CLICKBAIT	FULLNETCONC	**0.857**	**0.715**	**0.647**	**0.562**	**0.464**	**0.424**
	WENET	0.852	0.703	0.635	0.55	0.447	0.414
	LINGNET	0.82	0.64	0.528	0.391	0.224	0.171
	FULLNET	0.856	0.713	0.64	0.558	0.458	0.419

Table A.4: Performance of SOTA models after temperature scaling, with theoretical threshold (k_{fp} is set to 1)

TASK	MODEL	ACCURACY	VALUE$_1$	VALUE$_2$	VALUE$_4$	VALUE$_8$	VALUE$_{10}$
		($k_{fn} = k_{fp} = 0$)	($k_{fn} = 1$)	($k_{fn} = 2$)	($k_{fn} = 4$)	($k_{fn} = 8$)	($k_{fn} = 10$)
HATE-SPEECH	BADJ. ET AL., *Exp. 1*	**0.822**	**0.644**	**0.513**	**0.359**	**0.268**	**0.218**
	AGR. ET AL., *Exp. 1*	0.732	0.464	0.207	0.0	0.0	0.0
	BADJ. ET AL., *Exp. 2*	**0.489**	**-0.022**	0.0	0.0	0.0	0.0
	AGR. ET AL., *Exp. 2*	0.42	-0.16	0.0	0.0	0.0	0.0
CLICKBAIT	FULLNETCONC	**0.857**	**0.715**	**0.608**	**0.488**	**0.374**	0.331
	WENET	0.852	0.703	0.597	0.472	0.357	0.326
	LINGNET	0.82	0.64	0.499	0.348	0.173	0.115
	FULLNET	0.856	0.713	0.6	**0.488**	0.372	**0.335**

Table A.5: Performance of SOTA models after temperature scaling, with theoretical threshold ($k_{fp} = k_{fn}$)

.

TASK	MODEL	ACCURACY	VALUE$_1$	VALUE$_2$	VALUE$_4$	VALUE$_8$	VALUE$_{10}$
		($k_{fn} = k_{fp} = 0$)	($k_{fn} = 1$)	($k_{fn} = 2$)	($k_{fn} = 4$)	($k_{fn} = 8$)	($k_{fn} = 10$)
HATE-SPEECH	BADJ. ET AL., *Exp. 1*	**0.822**	**0.646**	**0.548**	**0.41**	**0.312**	**0.29**
	AGR. ET AL., *Exp. 1*	0.732	0.503	0.449	0.357	0.194	0.16
	BADJ. ET AL., *Exp. 2*	**0.489**	**0.077**	**0.077**	**0.077**	**0.077**	**0.077**
	AGR. ET AL., *Exp. 2*	0.42	0.0	0.0	0.0	0.0	0.0
CLICKBAIT	FULLNETCONC	**0.857**	**0.715**	**0.648**	**0.556**	**0.46**	**0.42**
	WENET	0.852	0.703	0.638	0.554	0.454	0.414
	LINGNET	0.82	0.639	0.535	0.381	0.232	0.188
	FULLNET	0.856	0.712	0.637	0.554	0.439	0.418

Table A.6: Performance of SOTA models, with empirical threshold (found on the validation set, k_{fp} is set to 1)

TASK	MODEL	ACCURACY	VALUE$_1$	VALUE$_2$	VALUE$_4$	VALUE$_8$	VALUE$_{10}$
		($k_{fn} = k_{fp} = 0$)	($k_{fn} = 1$)	($k_{fn} = 2$)	($k_{fn} = 4$)	($k_{fn} = 8$)	($k_{fn} = 10$)
HATE-SPEECH	BADJ. ET AL., *Exp. 1*	**0.822**	**0.646**	**0.539**	**0.38**	**0.259**	**0.228**
	AGR. ET AL., *Exp. 1*	0.732	0.503	0.405	0.313	0.151	0.117
	BADJ. ET AL., *Exp. 2*	**0.489**	**0.077**	-0.107	0.0	0.0	0.0
	AGR. ET AL., *Exp. 2*	0.42	0.0	**0.0**	0.0	0.0	0.0
CLICKBAIT	FULLNETCONC	**0.857**	**0.715**	**0.612**	0.488	**0.375**	**0.332**
	WENET	0.852	0.703	0.597	0.486	0.371	0.328
	LINGNET	0.82	0.639	0.51	0.34	0.179	0.134
	FULLNET	0.856	0.712	0.6	**0.489**	0.357	0.33

Table A.7: Performance of SOTA models with empirical threshold (found on validation set, $k_{fp} = k_{fn}$)

.

TASK	MODEL	ACCURACY	VALUE$_1$	VALUE$_2$	VALUE$_4$	VALUE$_8$	VALUE$_{10}$
		($k_{fn} = k_{fp} = 0$)	($k_{fn} = 1$)	($k_{fn} = 2$)	($k_{fn} = 4$)	($k_{fn} = 8$)	($k_{fn} = 10$)
HATE-SPEECH	BADJ. ET AL., *Exp. 1*	**0.822**	**0.653**	**0.552**	**0.422**	**0.328**	**0.297**
	AGR. ET AL., *Exp. 1*	0.732	0.503	0.452	0.363	0.225	0.16
	BADJ. ET AL., *Exp. 2*	**0.489**	**0.077**	**0.077**	**0.077**	**0.077**	**0.077**
	AGR. ET AL., *Exp. 2*	0.42	0.0	0.0	0.0	0.0	0.0
CLICKBAIT	FULLNETCONC	**0.857**	**0.716**	**0.649**	**0.565**	**0.468**	**0.438**
	WENET	0.852	0.706	0.638	0.555	0.456	0.428
	LINGNET	0.82	0.643	0.536	0.394	0.242	0.198
	FULLNET	0.856	0.714	0.641	0.559	0.46	0.429

Table A.8: Performance of SOTA models, with empirical threshold (found on the test set, k_{fp} is set to 1)

TASK	MODEL	ACCURACY	VALUE$_1$	VALUE$_2$	VALUE$_4$	VALUE$_8$	VALUE$_{10}$
		($k_{fn} = k_{fp} = 0$)	($k_{fn} = 1$)	($k_{fn} = 2$)	($k_{fn} = 4$)	($k_{fn} = 8$)	($k_{fn} = 10$)
HATE-SPEECH	BADJ. ET AL., *Exp. 1*	**0.822**	**0.653**	**0.542**	**0.399**	**0.284**	**0.243**
	AGR. ET AL., *Exp. 1*	0.732	0.503	0.408	0.32	0.182	0.117
	BADJ. ET AL., *Exp. 2*	**0.489**	**0.077**	0.0	0.0	0.0	0.0
	AGR. ET AL., *Exp. 2*	0.42	0.0	0.0	0.0	0.0	0.0
CLICKBAIT	FULLNETCONC	**0.857**	**0.716**	**0.613**	**0.497**	**0.384**	**0.351**
	WENET	0.852	0.706	0.599	0.487	0.373	0.343
	LINGNET	0.82	0.643	0.511	0.352	0.188	0.143
	FULLNET	0.856	0.714	0.603	0.493	0.379	0.344

Table A.9: Performance of SOTA models after temperature scaling, with empirical threshold (on test set, $k_{fp} = k_{fn}$)

MDS dataset - Supplementary Results

Model	Accuracy	Value				
		k=1	k=2	k=4	k=8	k=10
LogReg	0.74	0.48	0.375	0.295	0.255	**0.251**
MLP1	0.728	0.457	0.36	0.298	**0.26**	**0.251**
MLP4	0.72	0.439	0.327	0.16	-0.089	-0.193
mttri	0.753	0.506	0.352	0.072	-0.431	-0.663
T5	0.789	0.578	0.47	0.253	-0.179	-0.395
SieBERT	**0.836**	**0.672**	**0.577**	**0.392**	0.029	-0.15

Table A.10: Performance of models on Multi Domain Sentiment Dataset, TARGET = DVD, $k_{fp} = 1$, $k_{fn} = k$ (Value with theoretical threshold)

Model	Accuracy	Value				
		k=1	k=2	k=4	k=8	k=10
LogReg	0.74	0.48	0.283	0.122	0.038	0.027
MLP1	0.728	0.457	0.274	0.133	**0.054**	**0.038**
MLP4	0.72	0.439	0.202	-0.158	-0.737	-0.981
mttri	0.753	0.506	0.28	-0.123	-0.84	-1.166
T5	0.789	0.578	0.367	-0.056	-0.9	-1.323
SieBERT	**0.836**	**0.672**	**0.508**	**0.193**	-0.436	-0.747

Table A.11: Performance of models on Multi Domain Sentiment Dataset, TARGET = DVD, $k_{fp} = k_{fn} = k$ (Value with theoretical threshold)

Model	Accuracy	Value				
		k=1	k=2	k=4	k=8	k=10
LogReg	0.704	0.408	0.332	0.269	**0.222**	**0.219**
MLP1	0.691	0.382	0.272	0.197	0.18	0.183
MLP4	0.696	0.393	0.258	0.081	-0.183	-0.283
mttri	0.742	0.484	0.32	0.018	-0.52	-0.789
T5	0.77	0.541	0.468	0.321	0.029	-0.117
SieBERT	**0.826**	**0.652**	**0.564**	**0.389**	0.038	-0.131

Table A.12: Performance of models on Multi Domain Sentiment Dataset, TARGET = Books, $k_{fp} = 1$, $k_{fn} = k$ (Value with theoretical threshold)

Model	Accuracy	Value				
		k=1	k=2	k=4	k=8	k=10
LogReg	0.704	0.408	0.228	0.102	**0.022**	**0.015**
MLP1	0.691	0.382	0.134	0.013	-0.017	-0.013
MLP4	0.696	0.393	0.154	-0.171	-0.666	-0.86
mttri	0.742	0.484	0.254	-0.16	-0.869	-1.215
T5	0.77	0.541	0.311	-0.148	-1.066	-1.525
SieBERT	**0.826**	**0.652**	**0.479**	**0.136**	-0.547	-0.879

Table A.13: Performance of models on Multi Domain Sentiment Dataset, TARGET = Books, $k_{fp} = k_{fn} = k$ (Value with theoretical threshold)

Model	Accuracy	Value				
		k=1	k=2	k=4	k=8	k=10
LogReg	0.762	0.524	0.442	**0.355**	**0.293**	**0.282**
MLP1	0.749	0.497	0.413	0.338	0.284	0.274
MLP4	0.735	0.47	0.33	0.09	-0.313	-0.492
mttri	0.808	0.616	0.495	0.286	-0.085	-0.245
T5	0.784	0.568	0.417	0.116	-0.486	-0.787
SieBERT	**0.842**	**0.685**	**0.572**	0.349	-0.093	-0.314

Table A.14: Performance of models on Multi Domain Sentiment Dataset, TARGET = Electronics, $k_{fp} = 1$, $k_{fn} = k$ (Value with theoretical threshold)

Model	Accuracy	Value				
		k=1	k=2	k=4	k=8	k=10
LogReg	0.762	0.524	0.339	0.162	0.053	0.033
MLP1	0.749	0.497	0.327	0.18	**0.081**	**0.062**
MLP4	0.735	0.47	0.24	-0.143	-0.78	-1.06
mttri	0.808	0.616	0.441	0.148	-0.354	-0.58
T5	0.784	0.568	0.352	-0.08	-0.944	-1.376
SieBERT	**0.842**	**0.685**	**0.527**	**0.217**	-0.397	-0.705

Table A.15: Performance of models on Multi Domain Sentiment Dataset, TARGET = Electronics, $k_{fp} = k_{fn} = k$ (Value with theoretical threshold)

Model	Accuracy	Value				
		k=1	k=2	k=4	k=8	k=10
LogReg	0.782	0.565	0.466	0.365	0.306	0.295
MLP1	0.765	0.53	0.433	0.339	0.292	0.279
MLP4	0.761	0.521	0.416	0.263	0.026	-0.076
mttri	0.821	0.642	0.589	0.503	**0.376**	**0.31**
T5	0.777	0.555	0.427	0.172	-0.338	-0.594
SieBERT	**0.865**	**0.73**	**0.644**	**0.477**	0.147	-0.018

Table A.16: Performance of models on Multi Domain Sentiment Dataset, TARGET = Kitchen, $k_{fp} = 1$, $k_{fn} = k$ (Value with theoretical threshold)

Model	Accuracy	Value				
		k=1	k=2	k=4	k=8	k=10
LogReg	0.782	0.565	0.374	0.176	0.06	0.034
MLP1	0.765	0.53	0.337	0.164	**0.07**	**0.044**
MLP4	0.761	0.521	0.312	0.003	-0.478	-0.685
mttri	0.821	0.642	0.489	0.235	-0.192	-0.384
T5	0.777	0.555	0.332	-0.113	-1.004	-1.449
SieBERT	**0.865**	**0.73**	**0.595**	**0.328**	-0.195	-0.454

Table A.17: Performance of models on Multi Domain Sentiment Dataset, TARGET = Kitchen, $k_{fp} = k_{fn} = k$ (Value with theoretical threshold)

		Source			
		DVD	Books	Electronics	Kitchen
Target	DVD	-	**0.782**	0.716	0.723
	Books	**0.718**	-	0.694	0.7
	Electronics	0.731	0.723	-	**0.831**
	Kitchen	0.762	0.746	**0.839**	-

Table A.18: Accuracy of LogReg model on Multi Domain Sentiment Dataset.

		Source			
		DVD	Books	Electronics	Kitchen
Target	DVD	-	**0.761**	0.704	0.72
	Books	**0.711**	-	0.68	0.683
	Electronics	0.712	0.706	-	**0.828**
	Kitchen	0.743	0.722	**0.83**	-

Table A.19: Accuracy of MLP1 model on Multi Domain Sentiment Dataset.

		Source			
		DVD	Books	Electronics	Kitchen
Target	DVD	-	**0.763**	0.692	0.704
	Books	**0.714**	-	0.681	0.695
	Electronics	0.689	0.683	-	**0.832**
	Kitchen	0.74	0.715	**0.827**	-

Table A.20: Accuracy of MLP4 model on Multi Domain Sentiment Dataset.

		Source			
		DVD	Books	Electronics	Kitchen
Target	DVD	-	**0.808**	0.727	0.724
	Books	**0.753**	-	0.733	0.741
	Electronics	0.761	0.801	-	**0.862**
	Kitchen	0.828	0.777	**0.859**	-

Table A.21: Accuracy of mttri model on Multi Domain Sentiment Dataset

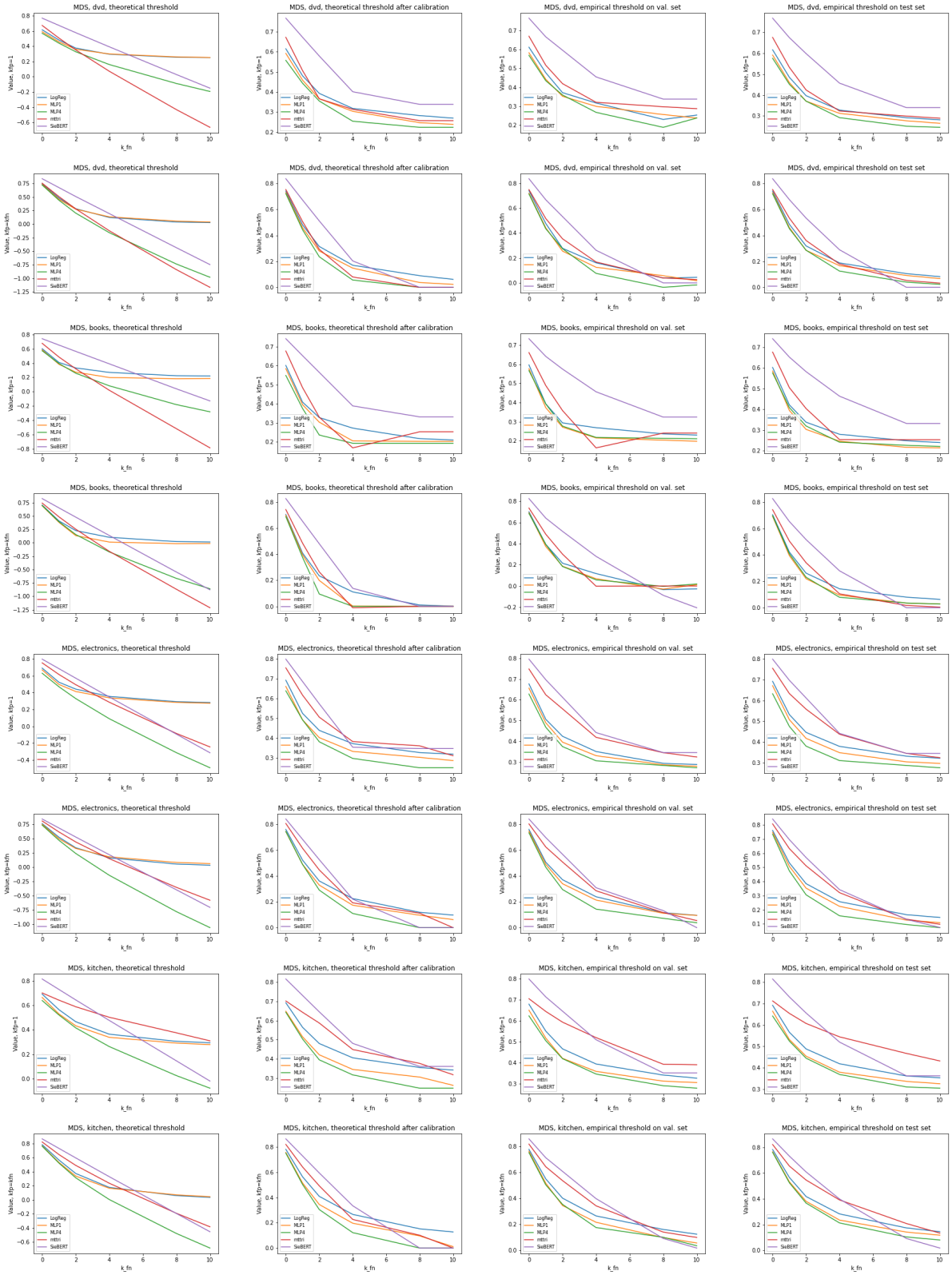


Figure A.2: Value curves on Multi Domain Sentiment dataset for increasing k_{fn}, Values are averaged for simple models, the 'mttri' model, and GPT-3.

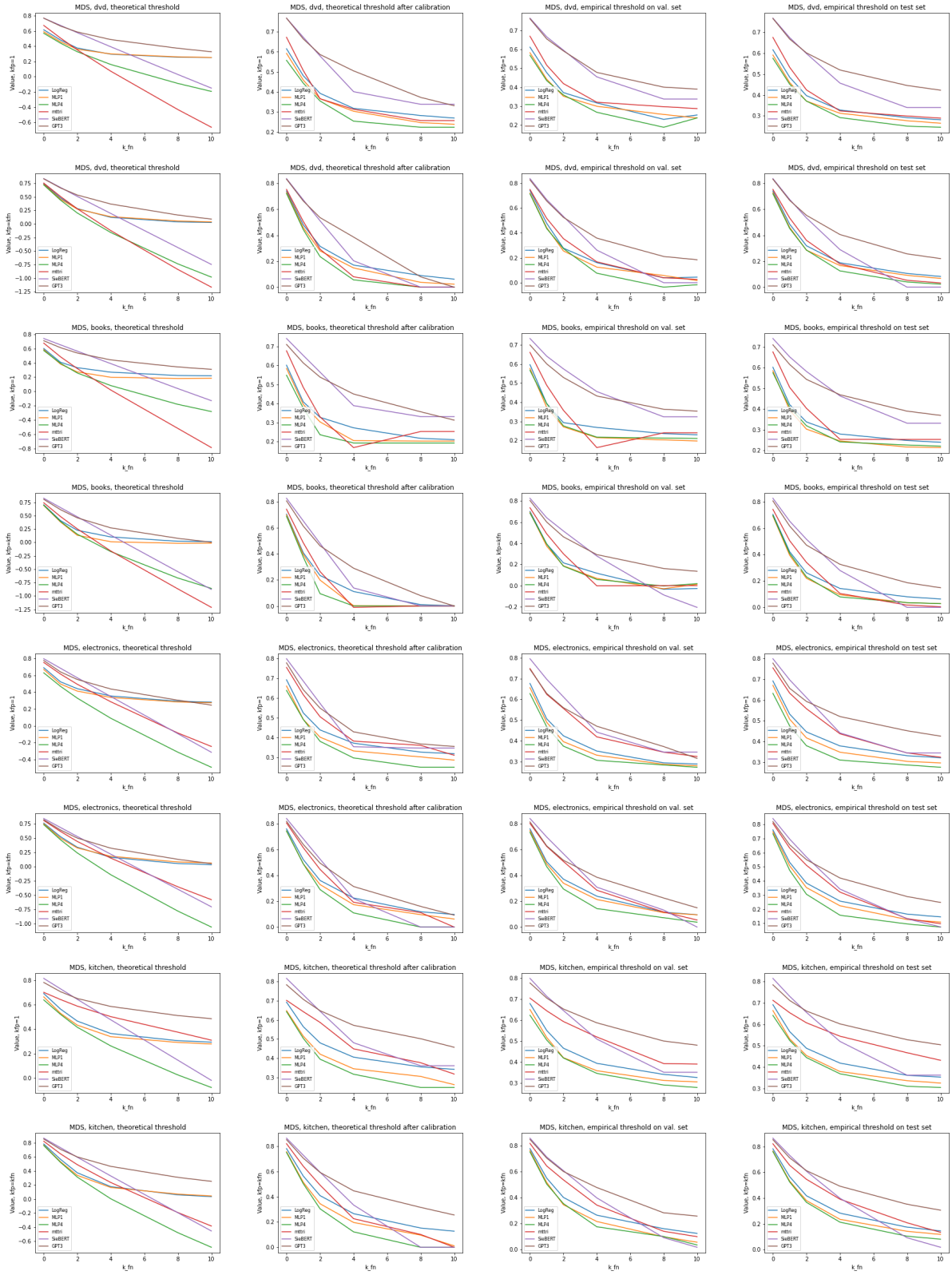


Figure A.3: Value curves on Multi Domain Sentiment dataset for increasing k_{fn}, Values are averaged for simple models, the 'mttri' model, and GPT-3.

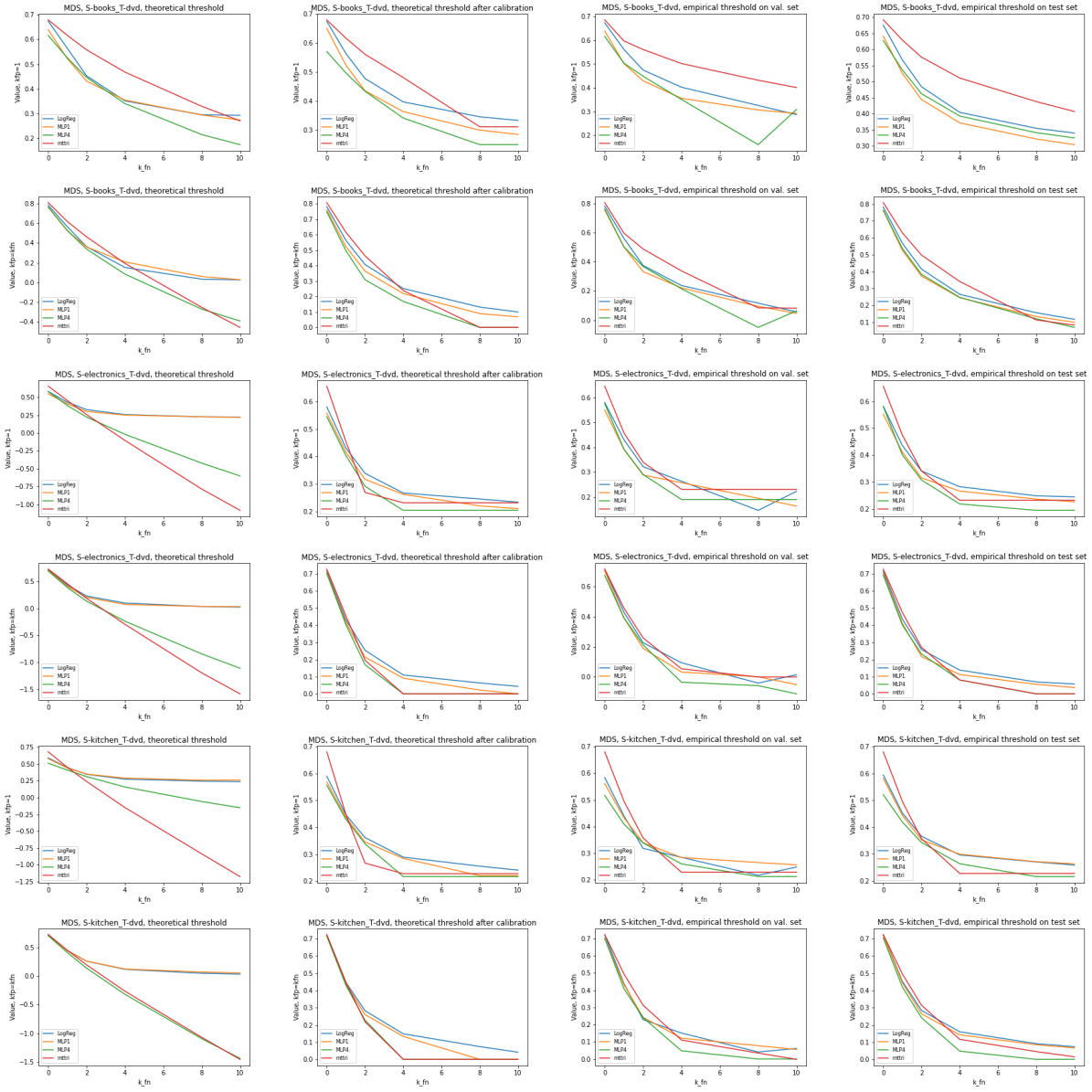


Figure A.4: Value curves on Multi Domain Sentiment dataset for increasing k_{fn}, TARGET = 'DVD'. We use S-SOURCE_T-TARGET format in the sub-figure titles, values show the performance of each model trained on a source domain and tested on a target domain.

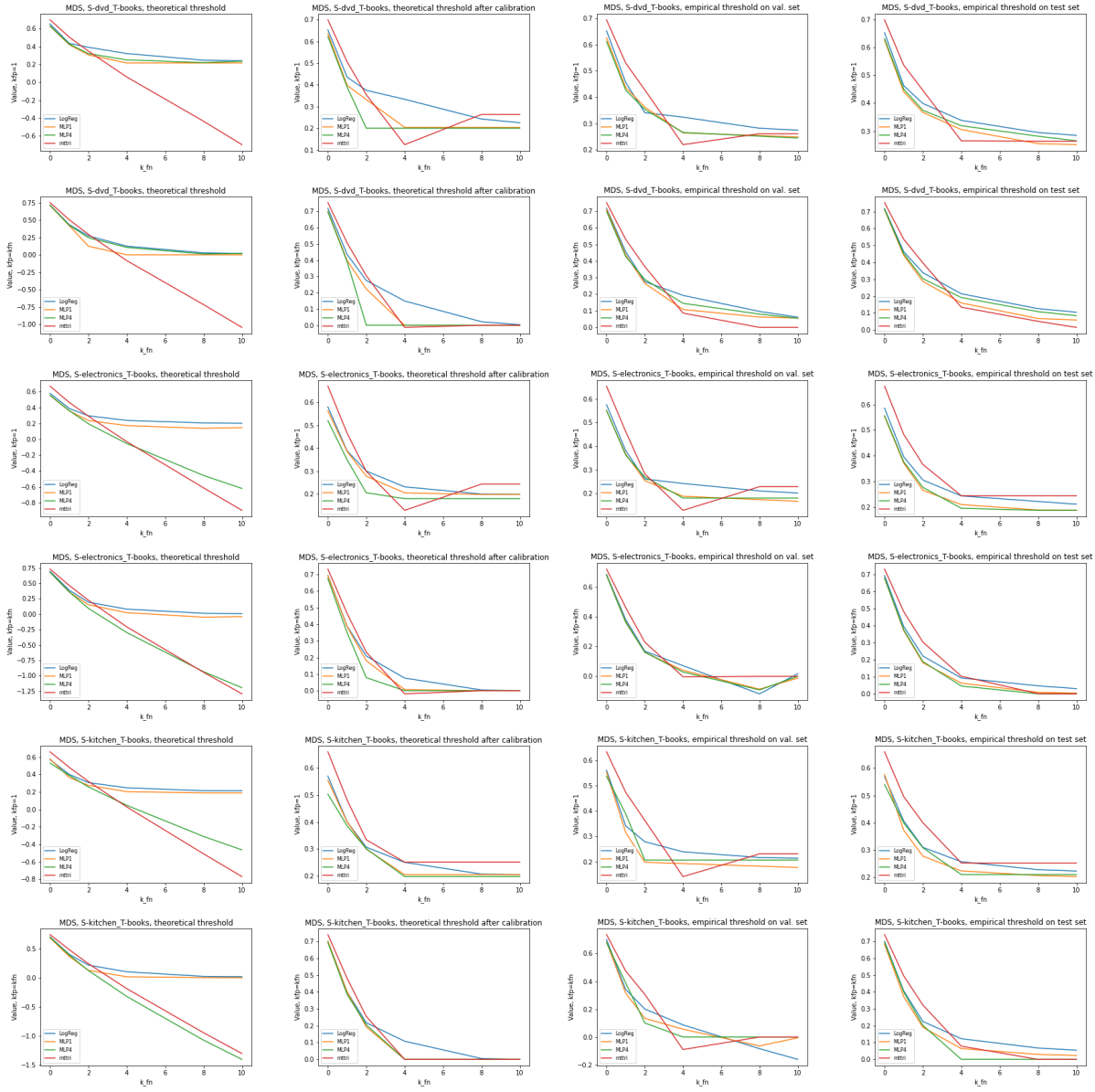


Figure A.5: Value curves on Multi Domain Sentiment dataset for increasing k_{fn}, TARGET = 'Books'. We use S-SOURCE_T-TARGET format in the sub-figure titles, values show the performance of each model trained on a source domain and tested on a target domain.

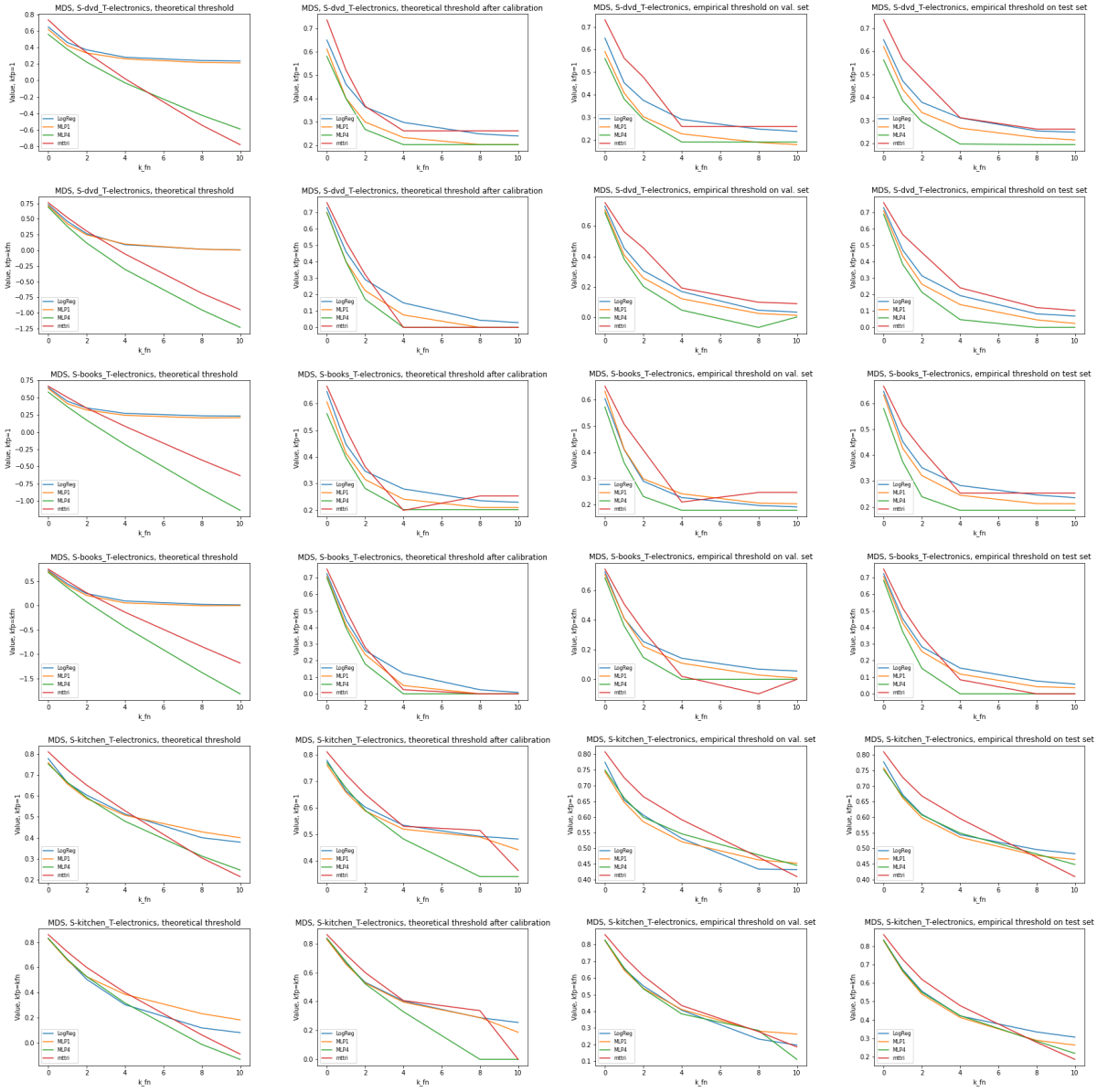


Figure A.6: Value curves on Multi Domain Sentiment dataset for increasing k_{fn}, TARGET = 'Electronics'. We use S-SOURCE_T-TARGET format in the sub-figure titles, values show the performance of each model trained on a source domain and tested on a target domain.

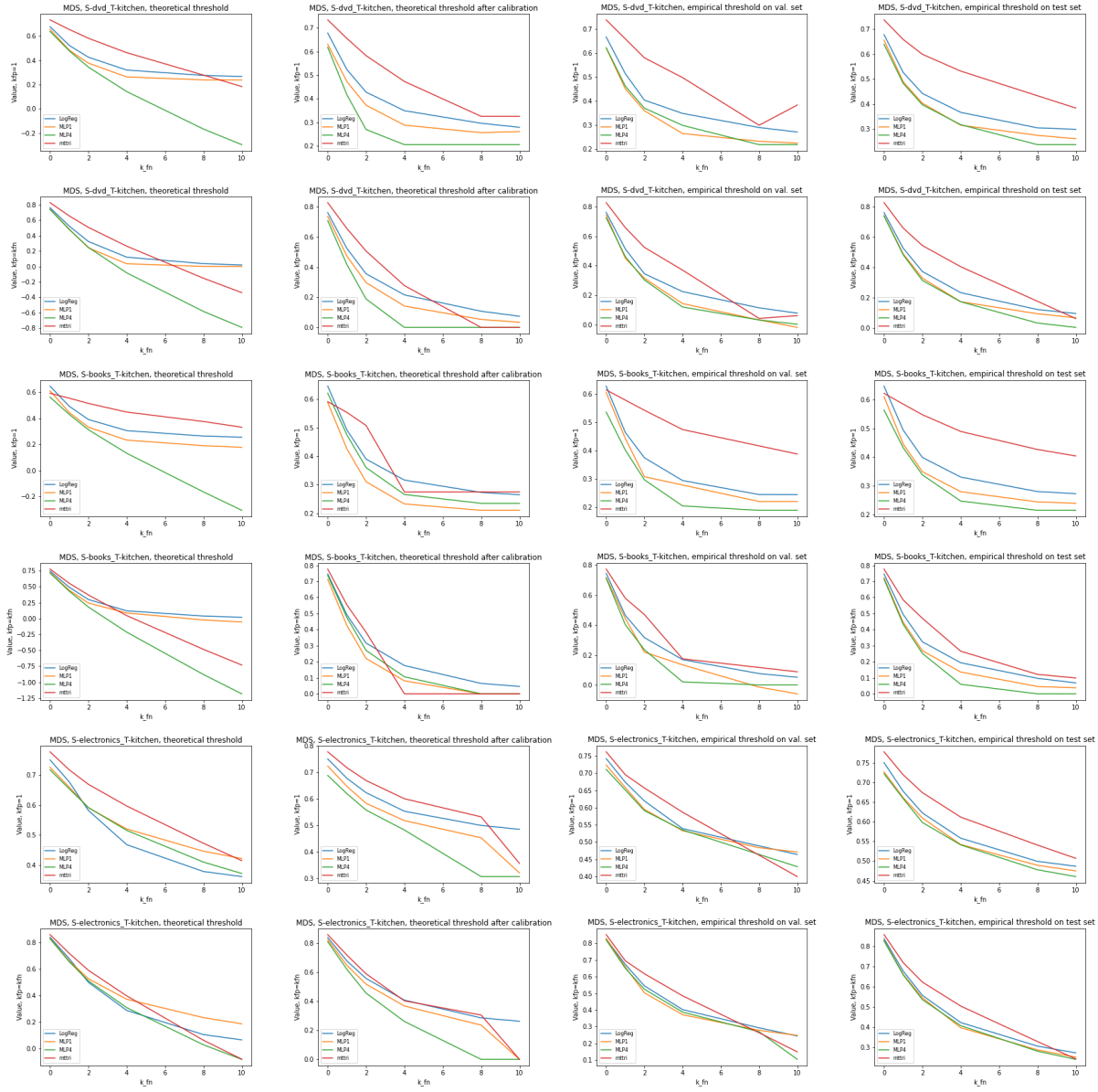


Figure A.7: Value curves on Multi Domain Sentiment dataset for increasing k_{fn}, TARGET = 'Kitchen'. We use S-SOURCE_T-TARGET format in the sub-figure titles, values show the performance of each model trained on a source domain and tested on a target domain.

Multi-class datasets - Supplementary Results

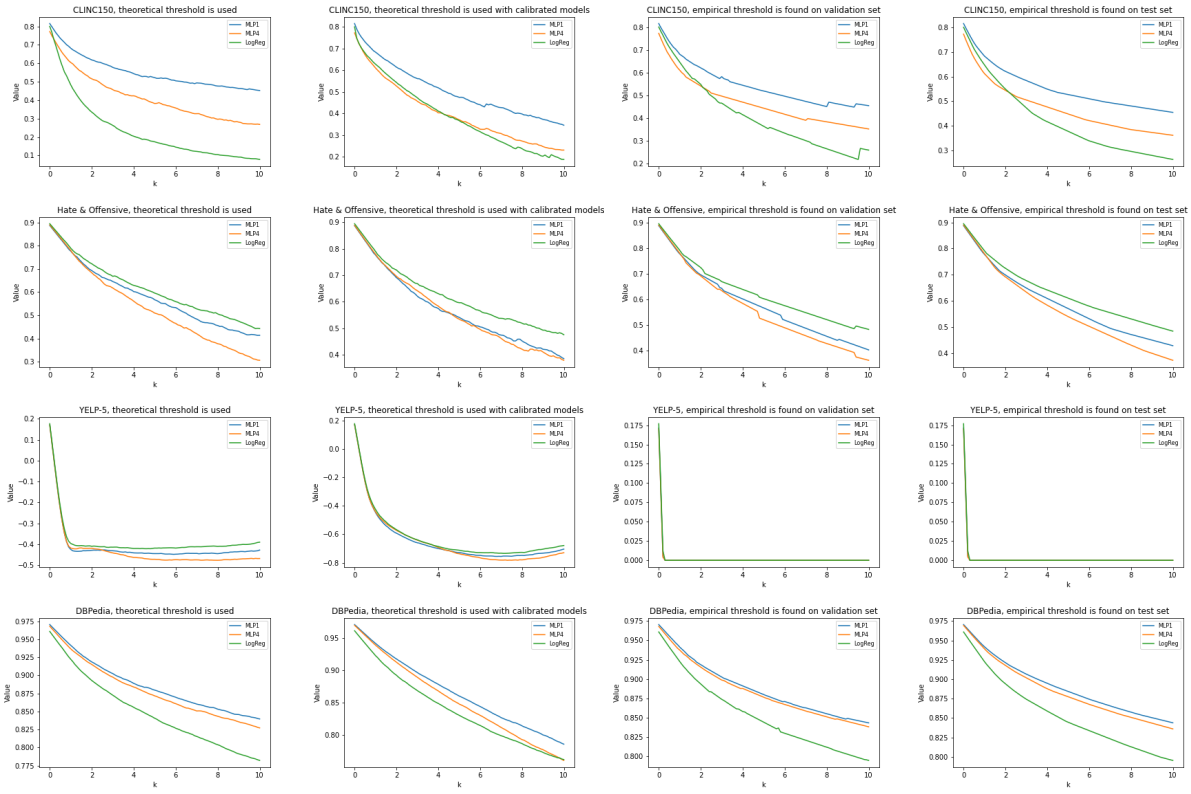


Figure A.8: Value curves of multi-class datasets for increasing k.

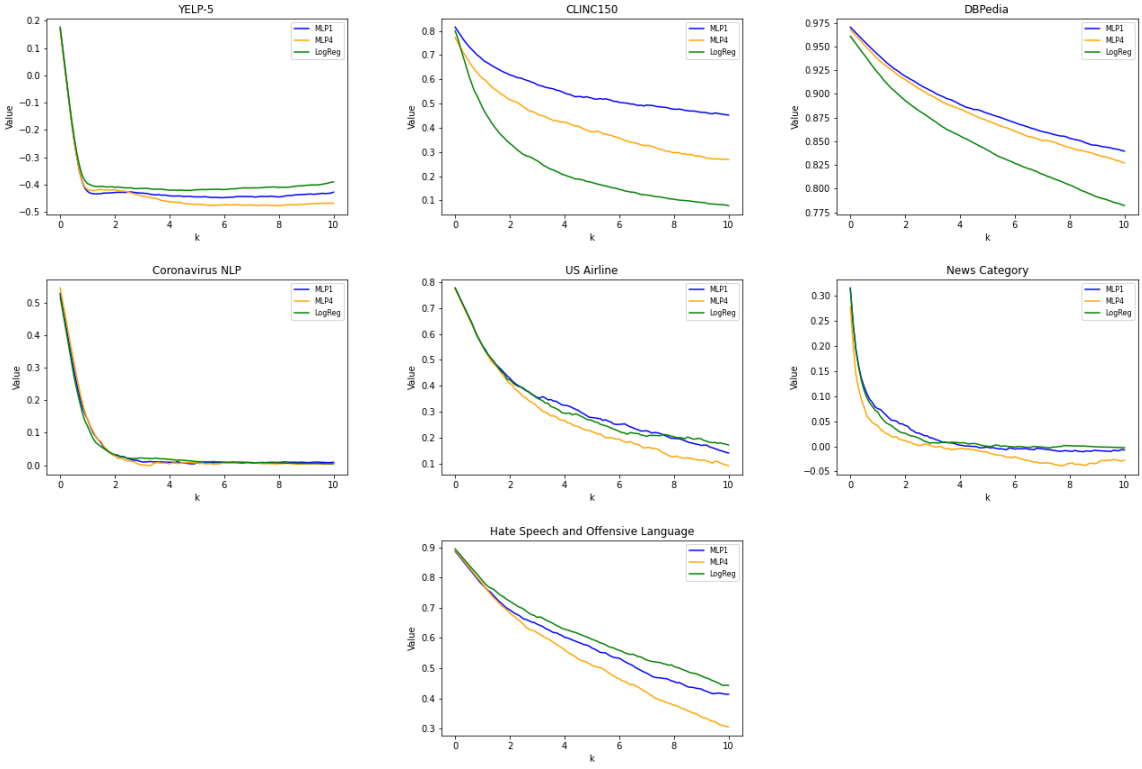


Figure A.9: Value curves with theoretical threshold for increasing k.

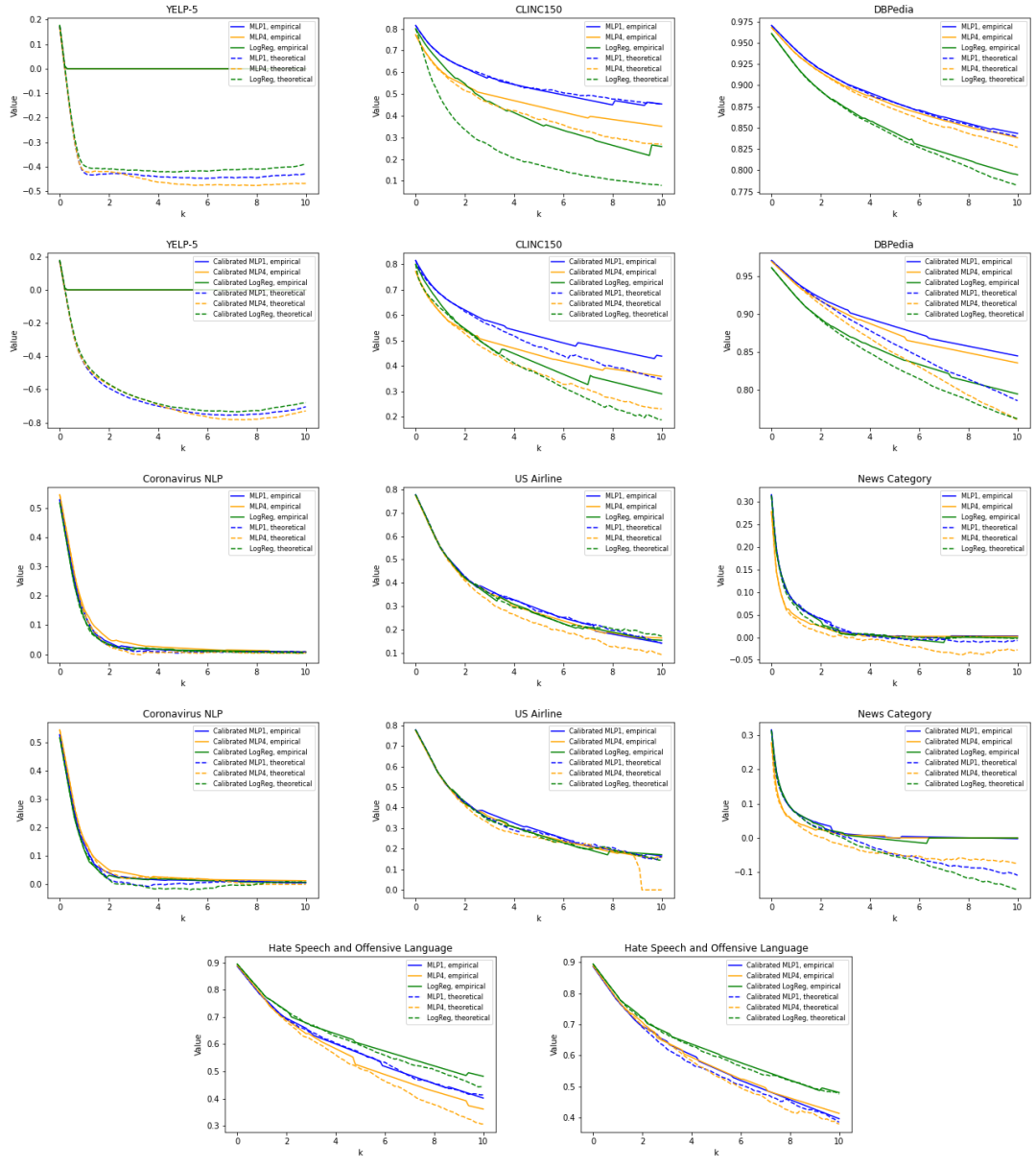


Figure A.10: Value curves with theoretical and empirical threshold (on the validation set) for increasing k.

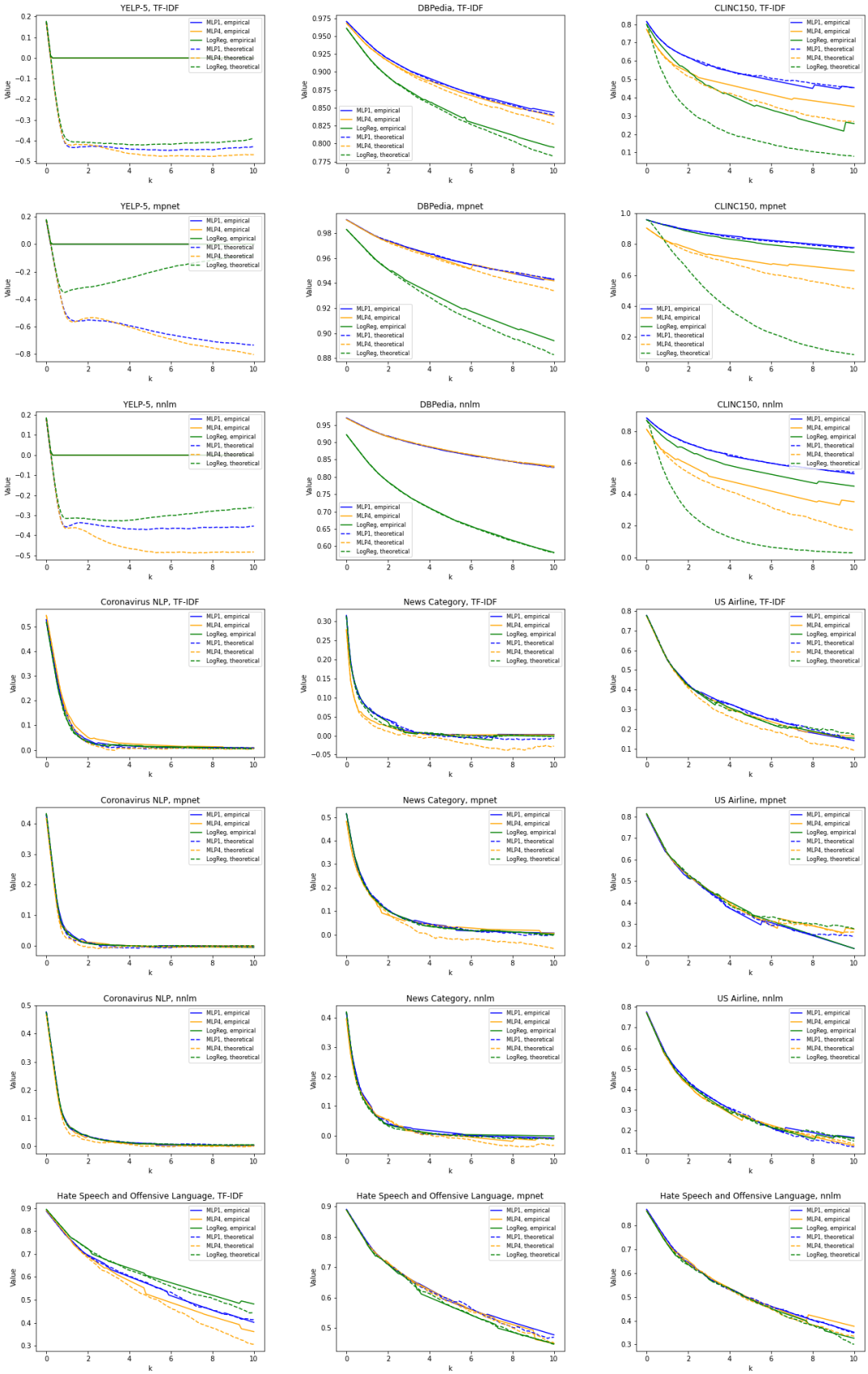


Figure A.11: Effect of using different text encoders. Value curves with theoretical and empirical threshold (on the validation set) for increasing k.

A.2 Supplemental material for Chapter 6

A.2.1 Results - Table with LogReg+MPNet results, using theoretical threshold

AL Strategy	Batch	US Airline				Clinc150				DBPedia				Hate Speech			
		Value				Value				Value				Value			
		κ=0	κ=2	κ=4	κ=8	κ=0	κ=2	κ=4	κ=8	κ=0	κ=2	κ=4	κ=8	κ=0	κ=2	κ=4	κ=8
Uncertainty	5	**0.8**	0.391	0.219	0.082	0.608	0.0	0.0	0.0	**0.933**	0.369	0.083	0.003	**0.87**	**0.641**	0.456	0.237
	10	**0.814**	0.451	0.298	0.148	**0.869**	0.0	0.0	0.0	**0.961**	0.566	0.212	0.037	**0.883**	0.659	0.495	0.297
	20	**0.815**	0.485	0.336	0.224	**0.943**	0.0	0.0	0.0	**0.971**	**0.776**	0.487	0.187	**0.885**	0.683	0.545	0.353
	30	0.819	**0.511**	0.362	0.271	**0.96**	0.0	0.0	0.0	**0.972**	**0.863**	**0.702**	0.452	**0.888**	**0.699**	0.578	0.411
Random	5	0.784	0.405	0.273	0.153	0.519	0.0	0.0	0.0	0.852	**0.502**	0.324	0.191	0.853	0.616	0.502	0.348
	10	0.795	0.455	0.314	0.22	0.716	0.0	0.0	0.0	0.899	**0.643**	**0.468**	0.304	0.868	0.642	0.532	0.401
	20	0.802	0.488	**0.362**	0.256	0.882	0.002	0.0	0.0	0.927	0.74	**0.605**	**0.434**	0.873	0.663	0.556	0.432
	30	0.805	0.502	0.365	0.278	0.919	0.015	0.0	0.0	0.942	0.79	0.672	**0.513**	0.877	0.676	0.571	0.453
Certainty	5	0.653	0.199	0.177	0.139	0.095	0.022	**0.018**	0.009	0.442	0.098	0.076	0.055	0.784	0.389	0.217	0.201
	10	0.639	0.112	-0.036	-0.003	0.121	**0.051**	**0.039**	**0.021**	0.443	0.078	0.071	0.073	0.781	0.371	0.11	0.036
	20	0.635	0.084	-0.079	-0.045	0.16	**0.104**	**0.076**	**0.03**	0.609	0.359	0.302	0.226	0.778	0.355	0.032	-0.193
	30	0.644	0.206	0.161	0.178	0.219	**0.153**	**0.11**	**0.048**	0.609	0.336	0.329	0.297	0.778	0.354	0.04	-0.206
TOS	5	0.796	0.373	**0.313**	**0.233**	0.284	0.015	0.013	**0.015**	0.909	0.394	0.333	0.064	0.855	0.631	**0.51**	0.341
	10	0.805	0.444	0.328	0.241	0.62	0.036	0.026	**0.021**	0.952	0.584	0.452	0.119	0.876	**0.671**	0.523	0.396
	20	0.813	0.493	0.355	**0.27**	0.9	0.075	0.057	0.027	**0.97**	0.742	0.557	0.148	**0.885**	**0.693**	0.567	**0.441**
	30	**0.822**	**0.511**	**0.376**	0.274	0.94	0.123	0.079	0.034	**0.971**	0.857	0.592	0.42	**0.888**	**0.697**	**0.588**	0.451
TOS-Below	5	0.485	**0.409**	0.307	0.213	**0.701**	**0.024**	0.009	**0.012**	0.342	0.451	**0.354**	**0.263**	0.216	**0.641**	0.503	**0.361**
	10	0.485	**0.46**	**0.331**	**0.249**	0.701	0.048	0.017	**0.022**	0.342	0.592	0.428	**0.35**	0.216	**0.671**	**0.536**	**0.41**
	20	0.485	**0.501**	**0.362**	**0.271**	0.701	0.085	0.051	**0.03**	0.342	0.746	0.525	0.415	0.216	0.691	**0.573**	**0.44**
	30	0.485	**0.512**	**0.378**	**0.291**	0.701	0.129	0.087	0.041	0.342	0.858	0.6	0.43	0.216	**0.699**	0.586	**0.461**

Table A.22: Performance of AL strategies with LogReg + MPNet (using theoretical threshold)

A.2.2 Results using empirical threshold

AL Strategy	Batch	US Airline				Clinc150				DBPedia				Hate Speech			
		Value				Value				Value				Value			
		k=0	k=2	k=4	k=8	k=0	k=2	k=4	k=8	k=0	k=2	k=4	k=8	k=0	k=2	k=4	k=8
Uncertainty	5	**0.719**	0.285	0.121	0.032	0.173	0.008	0.0	0.0	**0.835**	**0.586**	**0.468**	**0.329**	**0.796**	0.517	0.392	0.236
	10	**0.732**	**0.331**	0.171	0.076	0.451	0.063	0.008	0.0	**0.898**	**0.726**	**0.636**	**0.499**	**0.869**	0.64	0.494	0.303
	20	**0.752**	0.357	0.211	0.094	**0.665**	0.216	0.011	0.001	**0.934**	**0.82**	**0.754**	**0.676**	0.891	**0.702**	0.559	0.358
	30	**0.761**	0.381	0.253	0.143	**0.721**	0.296	0.054	0.003	**0.942**	**0.841**	**0.783**	**0.712**	**0.896**	0.703	0.581	0.419
Random	5	0.694	**0.29**	0.19	**0.095**	**0.405**	**0.043**	0.005	0.004	0.753	0.464	0.386	0.321	**0.795**	0.583	0.497	**0.386**
	10	0.72	0.313	**0.223**	**0.127**	**0.541**	**0.14**	0.031	0.006	0.834	0.601	0.53	0.467	0.83	0.602	0.521	0.418
	20	0.738	0.357	**0.258**	**0.167**	0.658	**0.279**	**0.119**	0.024	0.883	0.712	0.643	0.569	0.85	0.636	0.556	0.442
	30	0.748	0.367	**0.259**	**0.182**	0.704	**0.344**	**0.182**	0.059	0.905	0.761	0.693	0.625	0.865	0.659	0.58	0.466
Certainty	5	0.635	0.201	0.104	0.055	0.052	0.036	**0.034**	**0.034**	0.157	0.056	0.045	0.042	0.777	0.479	0.371	0.282
	10	0.631	0.256	0.168	0.095	0.075	0.053	0.052	**0.05**	0.143	0.086	0.083	0.078	0.778	0.49	0.405	0.317
	20	0.632	0.273	0.19	0.137	0.14	0.078	0.068	0.054	0.625	0.205	0.14	0.121	0.777	0.449	0.379	0.301
	30	0.635	0.296	0.212	0.147	0.207	0.105	0.084	0.062	0.436	0.197	0.176	0.155	0.777	0.422	0.323	0.23
TOS	5	0.689	0.244	0.145	0.05	0.141	0.005	0.016	0.026	0.794	0.344	0.285	0.076	0.787	0.579	0.454	0.249
	10	**0.732**	0.315	0.198	0.115	0.387	0.016	**0.03**	0.039	0.869	0.442	0.382	0.242	0.856	0.63	0.549	0.41
	20	0.746	0.348	0.214	0.153	0.643	0.045	0.053	0.057	0.93	0.634	0.416	0.379	**0.893**	0.688	0.574	0.485
	30	**0.762**	0.373	0.248	**0.182**	0.669	0.084	0.072	0.078	0.94	0.756	0.444	0.392	**0.896**	0.71	0.585	**0.503**
TOS-Below	5	0.255	0.276	**0.159**	0.084	0.361	0.013	0.009	0.024	0.283	0.396	0.325	0.154	0.388	**0.604**	**0.494**	0.374
	10	0.255	0.314	0.204	0.12	0.361	0.023	0.014	0.04	0.283	0.446	0.393	0.308	0.388	**0.659**	**0.558**	**0.454**
	20	0.255	**0.366**	0.217	0.122	0.361	0.043	0.042	**0.066**	0.283	0.648	0.423	0.387	0.388	**0.701**	**0.583**	**0.487**
	30	0.255	**0.388**	0.254	0.165	0.361	0.085	0.079	**0.085**	0.283	0.771	0.498	0.396	0.388	**0.72**	**0.595**	**0.503**

Table A.23: Performance of AL strategies with LogReg + TF-IDF (using empirical threshold tuned on test set)

A.2.3 Results - Performance of SOTA AL Strategies

Figure A.12 shows the performance of SOTA AL strategies over 4 multi-class datasets with Logistic Regression combined with TF-IDF or MPNet. While uncertainty sampling shows an outstanding performance in terms of accuracy, we observed that this is not the case when we change the metric to value and test with various cost settings. For high k values, we observed that the random sampling and in some cases even certainty sampling becomes better than uncertainty sampling in the early batches (when the model does not have enough data to learn the task). We show results for theoretical value where we assume the model is calibrated and the threshold is found based on the following equation: $t = k/(k+1)$.

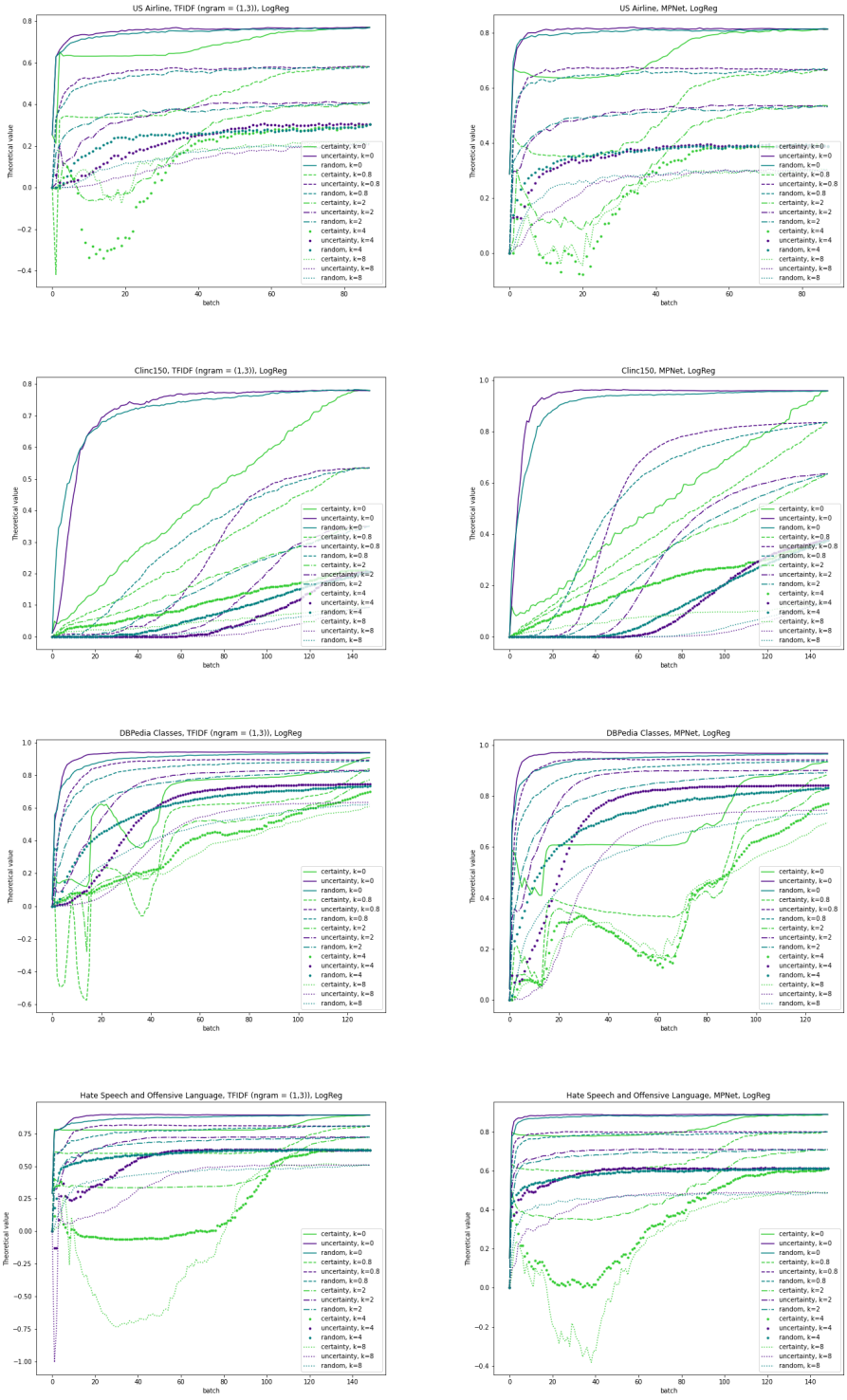


Figure A.12: Value curves with theoretical threshold for increasing number of AL batch. Model: Logistic Regression.

A.2.4 Results - TOS

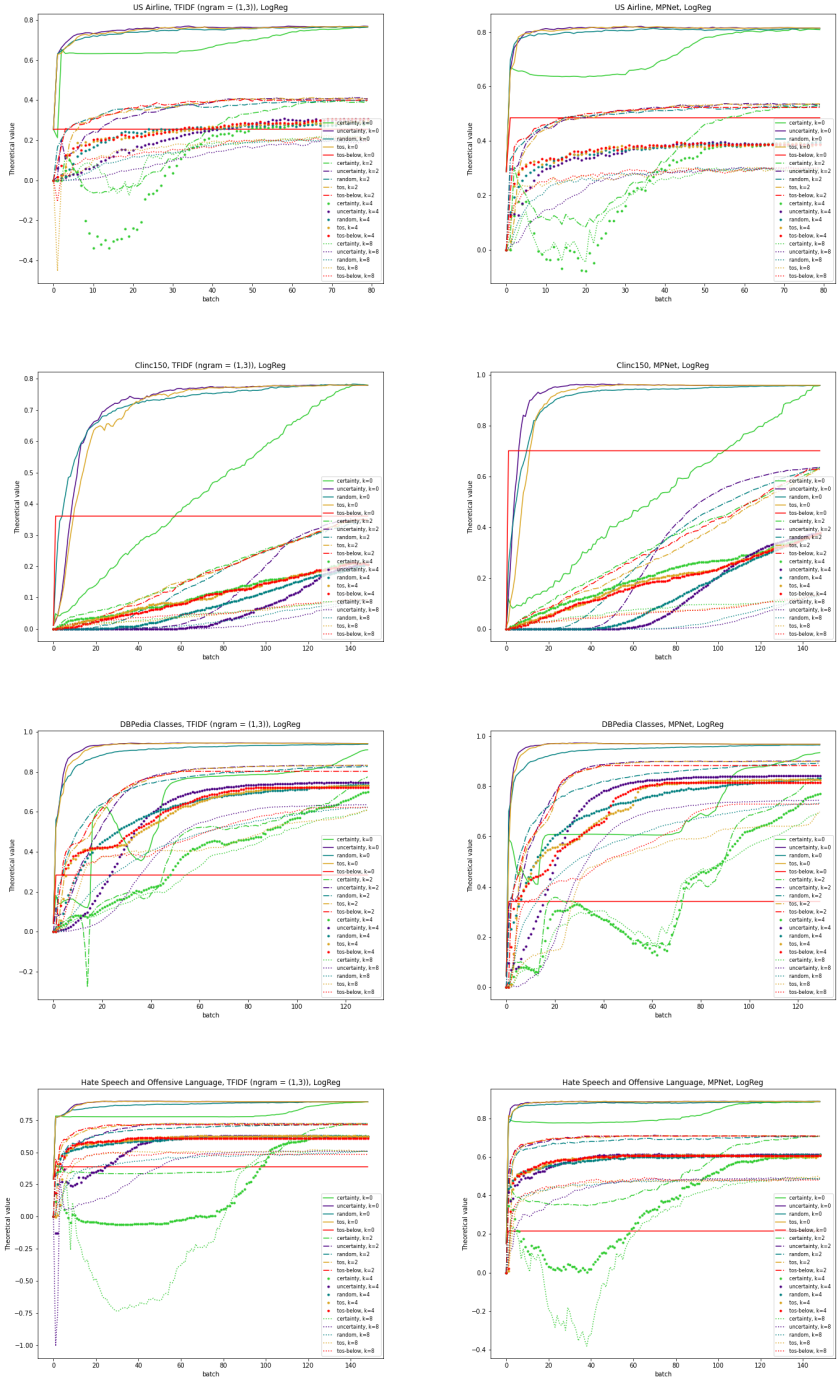


Figure A.13: Value curves with theoretical threshold for increasing number of AL batch. Model: Logistic Regression.

A.2.5 Results on News Category and Coronavirus Datasets

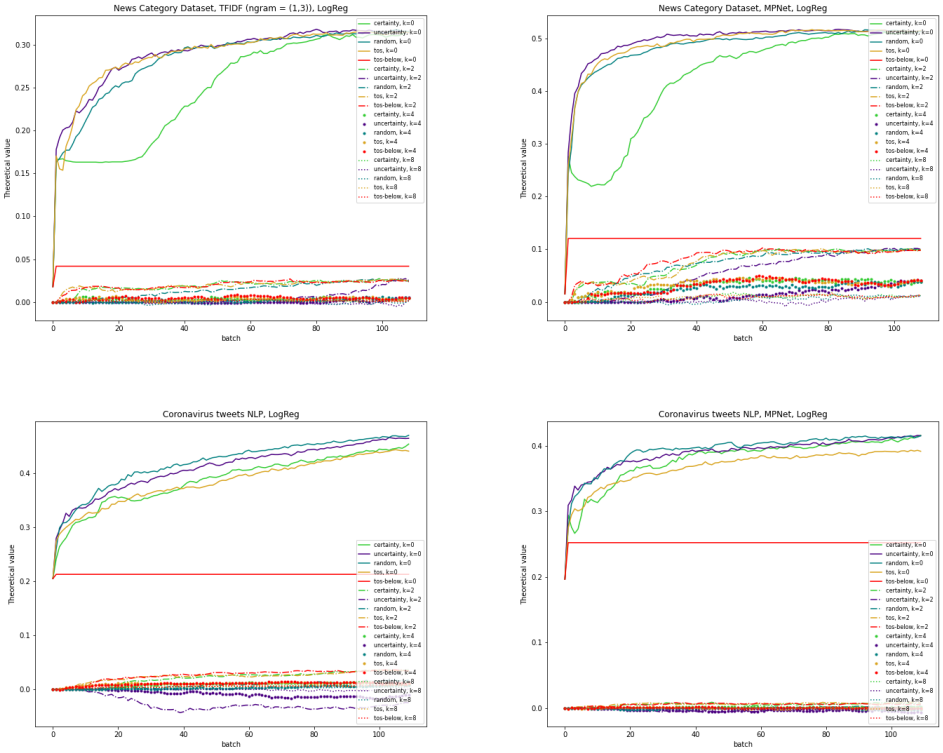


Figure A.14: Value curves with theoretical threshold for increasing number of AL batch. Model: Logistic Regression.

A.2.6 Results on US Airlines Dataset with fine-tuned RoBERTa

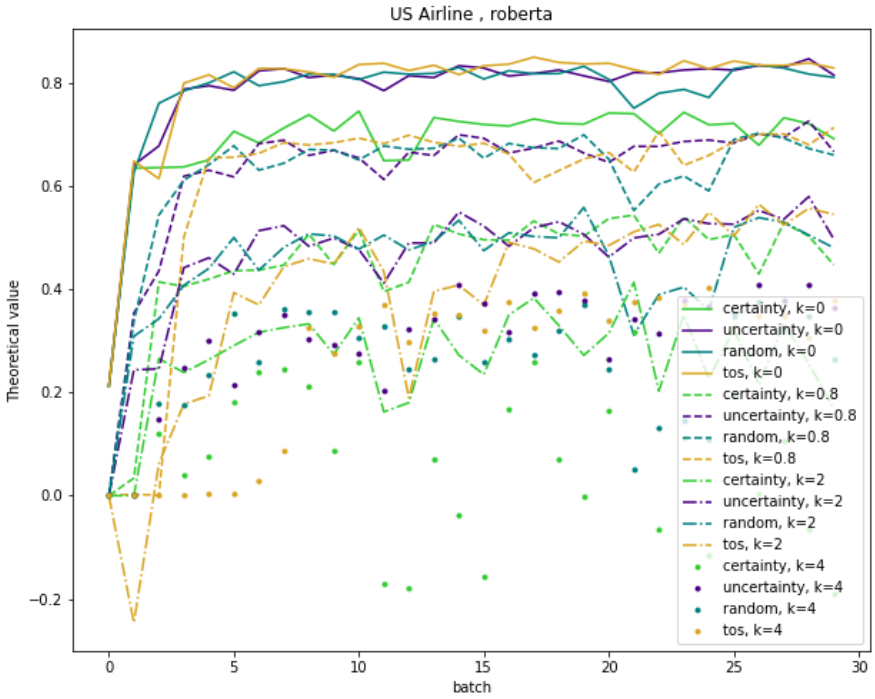


Figure A.15: Value curves with theoretical threshold for increasing number of AL batch. Model: Fine-tuned RoBERTa.