Calpric: Automated Classification of Privacy Policy Using Deep Active Learning and Crowdsourcing Techniques

by

Wendy (Wenjun) Qiu

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

© Copyright 2020 by Wendy (Wenjun) Qiu

Abstract

Calpric: Automated Classification of Privacy Policy Using Deep Active Learning and Crowdsourcing Techniques

Wendy (Wenjun) Qiu
Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto
2020

Privacy policies are statements that notify users of the services' data practices. However, few users are willing to read through policy texts due to the length and complexity. While there are existing privacy policy analysis tools based on machine learning, a large labeled training set is needed to achieve a high classification accuracy. Most existing policy corpora are labeled by skilled human annotators, requiring a significant amount of labor hours and effort. In this paper, we leverage active learning and crowdsourcing techniques to develop an automated classification tool named *Calpric* (Crowdsourcing Active Learning PRIvacy Policy Classifier), which is able to perform annotation equivalent to those done by skilled human annotators with high accuracy while minimizing the labeling cost. On average, our model is able to achieve the same F1 score using only 45.3% of the original labeling effort, and addresses the class imbalance issue of existing datasets.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor David Lie, for his invaluable advice and guidance throughout my research. I want to thank him for offering me this special opportunity and how much time and effort he spent on helping me with my work. This has been one of the most wonderful experience in my memory.

I am also really grateful for all the help and feedback I received from Professor Nicolas Papernot, Professor Gerald Penn, Professor Frank Rudzicz, Sean Robertson, Patricia Thaine and Professor Sepideh Ghanavati.

I would also like to thank all my research group members, especially Weicheng, Michelle and Mariana for their insightful ideas and feedback, and Shawn, Shirley, Gavin, John and Wei for all the fun we had.

Lastly, I would like to give my heartfelt thanks to my family and friends: my parents and my sister for supporting me throughout my study and my life in general. I would like to thank Cong, Xiaomeng and Peng for all the time we spent together. Thanks guys! I would also like to give a special thank to Shuai for being supportive, thoughtful and dependable through my ups and downs.

Contents

Chapter 1

Introduction

Privacy policies are legal documents that disclose how a party collects, uses, and shares private information data. Privacy legislation, such as the California Online Privacy Protection Act (CalOPPA) and the General Data Protection Regulation (GDPR), require online services to use privacy policies to obtain consent for collection and use of private information. However, studies have shown that users are unlikely to read privacy policies, as it would take hundreds of hours to read all the privacy policies a typical person encounters over a year [35]. Given this challenge, there have been many proposals to use machine learning-based text processing tools to distill critical information from privacy policies and provide it to both users and regulators [17, 62]. While there has been some success, an ongoing challenge with this approach is the difficulty of obtaining large, high-quality labeled training sets that are required by machine learning to be effective. For instance, Liu et al. [29] show that privacy policy paragraphs can be classified with average micro-F1 scores of 0.78, with their model being trained on the OPP-115 corpus [53], which consists of $23K$ data practices, $128K$ practice attributes, and $103K$ labeled text spans extracted from 115 privacy policies.

The difficulty of obtaining labeled data stems from the conventional wisdom that privacy policies are hard to read and understand, and thus must be labeled by skilled annotators. As an example, the most widely-used OPP-115 dataset is prepared by 10 skilled annotators (i.e law students or those with legal training), spending an average of 72 minutes on each privacy policy. While labeling privacy policies is labor intensive and expensive, unlabeled policies are easily accessible. In this work, we propose the use of active learning, in which a short series of sentences is extracted from the original privacy policy, where all sentences are related to the same data practice. An ideal segment contains all necessary information for labelers to understand the described data practices and excludes redundant details that may require extra time and effort to read.

To evaluate this idea, we design and implement *Calpric* (Crowdsourcing Active Learning PRIvacy Policy Classifier), which classifies a privacy policy as either collecting or not collecting three of the most commonly collected classes of private data: user contacts, user location and the device identifier. Although the number of categories in this thesis is limited, the machine learning model can be easily integrated and applied to other data categories, such as financial, demographic and health information. Calpric uses active learning to select the policy segments that will most improve model accuracy. These segments are sent to Amazon Mechanical Turk (mTurk) and labeled by crowdsourcing workers (Turkers).

Calpric is based on the Privacy Policy Word Embedding bidirectional Long Short Term Memory (PPWE-biLSTM) and focuses on labeling first party data practices involving collection/use of contact, location, and device information. We apply Calpric to a corpus of 52K privacy policies obtained by scraping the Google Play store, a large online market for Android applications. Using crowdsourcing and policy segmentation, along with careful design of the question survey and qualification tests, Calpric is able to capture policy segments with an average accuracy of 97.6%, which exceeds that of models trained on data sets labeled by skilled labelers (e.g., law students). We leverage **Active Learning** techniques to select labels that will improve model accuracy as much as possible while minimizing the labeling effort. Using such algorithms, classification models proactively select the most representative subset of available data to be labeled. The classifiers can be trained using a much smaller dataset while still achieving reasonably good performance, resulting in a more efficient learning procedure. We claim that Calpric is able to perform annotation equivalent to that done by skilled human annotators with high accuracy.

We also observe a side benefit from the usage of active learning. We find that one of the reasons Calpric is able to achieve significantly better accuracy is that active learning allows it to mitigate class imbalance, which is known to lead to lower model accuracy. Privacy policy labeling suffers from heavy class imbalance as there are significantly more positive segments that assert data collection (e.g., "The Application also accesses the names of individuals in the user's contacts."), than negative segments that assert the absence of data collection (e.g., "We do not collect your name, email addresses, postal addresses, and/or telephone numbers."). In fact, the most widely used labeled privacy policy datasets, OPP-115 and APP-350, contain only 2.0% and 18.6% negative segments, respectively. By selecting the policy segments that are most likely to improve model accuracy, we find that active learning allows Calpric to identify and use the negative samples in our large dataset and thus achieve a balanced training set despite the naturally occurring bias towards positive samples. An important note is that, this re-balancing does not reduce accuracy on true data. Further details are included in Chapter 5, where we evaluate Calpric on two sets of test data, including an imbalance test set which reflects the natural data distribution of privacy policies, and another set that is balanced using manual selection.

1.1 Contribution

In summary, this paper makes the following contributions:

1. We present Calpric, the first system we are aware of that applies active learning to the problem of privacy policy classification.

2. We examinate $375K$ Android app privacy policies from the Google Play Store, and prepare a corpus of $52K$ policies in raw text format, producing 153K, 63K, and 38K segments for contacts, location, and device ID categories, respectively.

3. We find that by automatically decomposing privacy policies into segments, Calpric is able to use active learning to identify the individual segments across privacy policies that are most valuable, allowing it to increase its accuracy with far fewer labeled segments.

4. We find that Calpric is able to automatically resolves class imbalance issue with privacy policy segments using active learning.

5. We study different design options in deploying Calpric and show that Calpric is able to achieve 97.6% labeling accuracy with Amazon Turker workers, which exceeds that of previous studies that used skilled workers to label policies.

1.2 Thesis Structure

We begin by giving relevant background knowledge in Chapter 2. We then summarize previous works on privacy policy classification and active learning in Chapter 3. We describe the detailed design and implementation of Calpric in Chapter 4, including Section 4.1 on data preparation, Section 4.2 on crowdsourcing and Section 4.3 on classification and active learning. Evaluation results are presented in Chapter 5. Finally, we discuss limitations and possible future plans and conclude our work in Chapter 6.

Chapter 2

Background

In this section, we introduce background knowledge on several topics related to our project. As mentioned, Calpric downloads and preprocess privacy policy texts, obtain labels from the crowdsourcing platform, and perform classification using machine learning networks. We describe the source of data collection, Google Play Store, and the crowdsourcing framework, Amazon mechanical Turk, in Section 2.1 and 4.2, respectively. We then provide a general introduction to natural language processing and neural networks in Section 2.3 and 2.4. Finally, we present the details of evaluation metrics selected to measure the performance of our tool in Section 2.5.

2.1 Google Play Store and Privacy Policy Texts

As the largest mobile application market, Google Play allows users to browse and download applications developed with the Android software development kit and published through Google [4]. Since 2017, Google Play requires each application should have with its own privacy policy [5]. We therefore select Google Play as the source platform to collect privacy policy texts. Unfortunately, there are also organizations that do not include any explicit statements disclosing collection and usage of user information [48]. We need to take into consideration when preparing the privacy policy corpus. In most cases, the available privacy policies are presented in the application and on the application homepage through an external link [5]. The website URL usually resolves to privacy policies in Hypertext Markup Language (HTML), Portable Document Format (PDF), or plaintext format. This project focuses on HTML webpages only. For our purpose, we extract the actual policy text from HTML pages, which is the standard markup language for documents designed to be displayed in a web browser [2]. The language is usually assisted by additional technologies, including Cascading Style Sheets (CSS)[1] and scripting languages such as JavaScript[2]. HTML defines the structure of a web page, including texts, images, and the apperance of the document. A website consists of a series of HTML elements, with the actual content embedded between a starting tag and an ending tag. For instance, a title named "HTML Elements Reference" should be written as *<title>HTML Elements Reference </title>* in the HTML document. Other text structural semantics include paragraphs, lists, links, quotes and so on, while additional tags such as

[1] `https://www.w3.org/Style/CSS/Overview.en.html`
[2] `https://www.javascript.com/`

<*img*> and <*table*> also exist. When converting HTML into the actual website, browsers use these tags to interpret the content of the page, but will not display the tags themselves.

When collecting privacy policies texts, we need to sanitize and process the HTML files by removing HTML tags and interactive elements. Details are included in the Section 4.1.

2.2 Crowdourcing

Crowdsourcing is a sourcing model in which a group of participants work together to achieve a common goal or a cumulative result, for instance ideas gathering, voting, micro-tasks and so on. The definition of crowdsourcing was first introduced in 2006 [20], and has become a crucial tool for businesses and organizations to use in various areas, such as data collection, human research and problem solving. In order to perform crowdsourcing, it is necessary to divide the project into smaller individual tasks. The crowdsourcers will then be assigned to different tasks and tackle the problems in parallel, making the process more efficient. Eventually, all results will be collected from the workers and go through review or post-processing if needed. Some example forms of crowdsourcing include Wikipedia[3] where the products a and LEGO Ideas, which allows users to share new ideas for potential Lego sets to be turned into commercial products [3].

2.2.1 Amazon Mechanical Turk

Modern crowdsourcing usually relies on a digital crowdsourcing platform, where crowdsourcers are united into one place and work on the micro-tasks. One of the most popular online crowdsourcing services is the Amazon Mechanical Turk (mTurk)[4]. Businesses and organizations known as *requesters* hire remotely located crowdworkers named *Turker* to perform discrete on-demand tasks that computers are currently unable to do. Jobs posted on mTurk is known as Human Intelligence Tasks (HITs). Turkers browse through existing HITs, select tasks and complete them in exchange for a payment pre-set by the requesters. Simple HITs can be posted on the mTurk requester website, which provides a UI interface to upload questions and format surveys. An open application programming interface (API) for mTurk is also available for posting more complex questions [1]. To publish new HITs, requesters need to perform two steps: creating projects and publishing batches.

Firstly, create a new project by providing a template, which is a design layout of the requesting jobs used by a specific group of HITs. The template is written in HTML, with CSS and scripting languages available for additional features. MTurk provides various customizable templates for commonly requested tasks, including survey, survey links, image classification, sentiment analysis and so on. Each project is associated with a project title, task description for workers to decide if they should take this or not, and keywords for workers to search for the task. Requesters also need to specify reward per assigment, which is the payment to each Turker to complete one HIT. There are also pre-set worker requirements for requesters to filter out unwanted groups of workers and select the most suitable ones. For instance, requesters can limit the HIT access to workers who are in specific regions, who speak a specific languages, and are in a specific range of age. Requesters can further select workers who perform well in previous

[3] `https://www.wikipedia.org/`
[4] `https://www.mturk.com/`

tasks, for example, *HIT Approval Rate for all Reuqesters'HITs* >85 and *Number of HITs Approved* >100.

As the second step, requesters will create new batches with existing projects by uploading a CSV file for each HIT to be published. The files contain the content of the actual questions, which should be different one batch to another. These data are inserted into the template created for this project.

Requesters are able to download all results in CSV once the Turkers finish their tasks. The task submission can be either approved or rejected by the requesters, and Turkers only receive payments on approved answers. Rejected Turkers will receive a notification from mTurk, and have the opportunity to communicate with requesters regarding the issue.

2.3 Natural Language Processing

Natural Language Processing (NLP) is a field of artificial intelligence that focuses on the interactions between machines and human languages. Typical examples of NLP challenges include speech recognition, text classification and language generation. Some commonly researched topics in natural language understanding are shown as follow:

Morphology: Morphology is the study of words. It identifies the basic building blocks of words and phrases, and analyze their structure and parts, such as prefixes, suffixes, stem and root words. For example, in most cases, "bad", "badly" and "worse" share a similar meaning. It also studies how context can be changed due to words' pronunciation and meaning.

Syntax: Syntax studies the rules and arrangement of words and phrases to form well-formed sentences in a language. One of the major tasks for this topic is to identify the parts of speech for each element in a sentence. For instance, the three words in "She loves him" associate with subject, verb and object, respectively.

Semantics: Semantics is the study of meaning in words and languages. There are various theories in linguistic semantics, including formal semantics, lexical semantics and cognitive semantics.

NLP systems are designed using different methods. One of the earlier algorithms is to assign hand-coding rules, such as by writing grammars and devising stemming heuristics [55]. Recent studies shift the focus to machine learning and statistical models, in which large corpora are analyzed and the NLP models are developed based on these real-word samples. Such methods have the following advantages over hand-coding rules:

1. Machine learning algorithms allow transfer learning, which is storing storing knowledge in one area and applying it to a different but related problem. For instance, a language model in English may be fine-tuned and used as a base model for sentiment analysis or privacy policy classification. However, hand-coding rules are usually designed to target a specific topic, which can be hardly re-used in other areas.

2. In the cases where input errors, such as misspelled words, and unfamiliar instances, such as words and phrases that the model never encounters during its training, rule-based systems suffer from significant difficulty in developing soft rules for unseen cases. In contrast, statistical models and machine learning systems are able to address such issues using statistical inference and techniques that designed to handle Out-Of-Vocabulary (OOV) words.

3. The accuracy of machine learning systems can be improved by introducing more training instances, whereas the rule-based systems can only do so by adding more rules, which increases the model complexity. It is also difficult to develop precise rules and manage the handcrafted heuristics. In general, machine learning based models are less time-consuming and considered as a more efficient alternative than rule-based models.

2.4 Neural Networks

A neural network is a circuit of nodes designed for solving artificial intelligence (AI) problems [19]. *Neurons* are nodes in the network which pass input values through functions and output the results. The connections of neurons are called *weights*, which carry values between each nodes. A simple feed-forward neural network consists of three layers: the input layer, the hidden layer(s) and the output layer. An activation function at the end of the layers controls the amplitude of the result. Common examples include Sigmoid, Tanh and Rectified Linear Unit (ReLU).

The training of neural networks involves the following steps: finding weights, calculating losses and gradient descent. Gradient descent is an iterative optimization algorithm for finding a local minimum of a differential function. Note that the training process is non-convex, meaning that the model may run into many local optima. To find model weights, we perform a forward pass in the neural network, traversing through all neurons from the first to the last layer; we then perform a backward pass known as back-propagation, where we compute the gradient in weight space of a feed-forward neural network with respect to a loss function. The algorithm updates the model weights by taking a step proportional to the negative of the gradient using the pre-set learning rate.

2.5 Evaluation Metrics

Calpric consists of three binary biLSTM classifiers. For performance evaluation, we focus on the following metrics: accuracy, precision, recall, F1 and the Matthew Correlation Coefficient (MCC), while both are calculated based on the confusion matrix.

2.5.1 The Confusion Matrix

True Positives (TP): Labels that are correctly predicted as positive ones by the classifier.
True Negatives (TN): Labels that are correctly predicted as negative ones by the classifier.
False Positives (FP): Labels that are predicted as positive, whereas it is in fact a negative label. The prediction contradicts with the actual class.
False Negatives (FN): Labels that are predicted as negative, whereas it is in fact a positive label. The prediction contradicts with the actual class.

2.5.2 Basic Metrics Calculation

To evaluate classification model performance, we calculate and compare the following metrics:
Accuracy: Accuracy is measured by the ratio of correct predictions to the total predictions. It is an indication of how often the classifier predicts correctly.

Accuracy = (TP + TN) / (TP + FP + FN + TN)

Precision: Precision is the ratio of correct positive predictions to the total positive predictions. It indicates how often the classifier makes the correct prediction given its prediction being positive.

Precision = TP / (TP + FP)

Recall (Sensitivity): The ratio of correct positive predictions to the all positive actual labels. It is the measurement of a probability that a relevant document is retrieved by the query.

Recall = TP / (TP + FN)

F1 score: F1 Score is the weighted average of Precision and Recall. It is a useful indication for situations where uneven class distribution occurs.

F1 Score = 2 * (Recall * Precision) / (Recall + Precision)

2.5.3 MCC Calculation

Desipte being the most frequently used metrics for classification models, there are some issues with the above four metrics: accuracy is sensitive to data imbalance; precision, recall and F1 are asymmetric in most cases. For instance, F1 is symmetric only when $TN = TP$ or $TN = 1 - TP$. To address these issues, we introduce the MCC score as an additional metric. It is equivalent to the mean square contingency coefficient, represents the correlation between target and predictions. MCC is calculated as follow:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}, \tag{2.1}$$

where TP, TN, FP, FN are true positives, true negatives, false positives and false negatives, respectively.

In binary classification, MCC is more informative since it takes into account the balance ratios of the four confusion matrix categories, especially when class imbalance issues [10] exist. The value varies between -1 and $+1$, where 1 is a perfect agreement between the actual and predicted labels, -1 is a perfect disagreement, and 0 occurs when the predictions are random with respect to the actuals.

Chapter 3

Related Work

We review existing privacy policy datasets in Section 3.1 and study the reliability of crowdsourcing labels in Section 3.2. Prior works on automated privacy policy analysis are analyzed in Section 3.3. We investigate active learning techniques and their application on top of deep learning models in Section 3.4.

3.1 Privacy Policy Datasets

Previous privacy policy datasets are mainly collected using two different approaches: policies labeled by skilled labelers, and policy labels created via crowdsourcing techniques. We study existing corpora and compare them with our data collection method.

Wilson et al. [53] compile a corpus of 115 website privacy policies, annotated with detailed information about the data practices that they describe, all produced by skilled annotators. This information consists of 23K data practices, 128K practice attributes, and 103K annotated text spans. Their annotation scheme consists of ten data practice categories, including *First Party Collection/Use*, *Third Party Collection/Sharing*, *User Choice/Control*, *User Access, Edit, & Deletion*, *Data Retention*, *Data Security*, *Policy Change*, *Do Not Track*, *International & Specific Audiences*, and *Other*. Under each categories, there are different attributes associated with the annotation. As an example, "Users are prompted to enter their emails before joining our community" should be categorized as a *First Party Collection/Use* on *contact* information, which is the sub-categorical attribute label. While being the most commonly used privacy policy dataset, OPP-115 does suffer from various problems, including the data imbalance issue where one category has significantly different number of labels than that of others.

Zimmeck et al. [61] contribute to the creation of a privacy policy corpus specifically for mobile apps, namely the APP-350 Corpus, using a similar annotation approach. Because mobile privacy policies usually involves less information than website-based policies, APP-350 develops a different annotation scheme: each annotation consists of a practice and a modality. For example, "We do not collect or use any location information from users" is labeled as *Location_data* and *NOT_PERFORMED*. Mobile privacy policies also suffer from class imbalance issue. To compensate the rarity of negative annotation labels, APP-350 introduces synthetic data by manually changing positive policy texts into negative samples.

Another corpus created by Elisa et al. [11] is extracted from 64 privacy policies. In this dataset, each

paragraph is manually annotated, resulting in a total of 1049 annotated paragraphs. The labeling is performed by a single annotator, but an additional agreement test is also performed to ensure the quality. While the above data corpora only focus on generating privacy policy datasets using legal experts or skilled annotators, we leverage crowdsourcing techniques as it is a more efficient and scalable approach to generate a relatively large sample set.

There are also prior works related to label collection with crowdsourcing. Zimmeck et al. [60] present a software architecture for analyzing essential policy terms based on crowdsourcing and automatic classification techniques. The tool, Privee, uses ToS;DR as the crowdsourcing database, which has not gained popularity and has a very limited collection of privacy policies. Instead, we use Amazon Mechanical Turk, which is the most popular crowdsourcing platform to perform discrete on-demand tasks. mTurk also has its own APIs for requester to implement additional features such as designing qualification tests.

When creating privacy policy labels, Wilson et al. [54] develop an annotation tool to enable both crowdworkers and skilled annotators to annotate privacy policies online. However, their system requires labelers to read through the entire privacy policy and select specific texts to annotate. We, instead, preprocessed privacy policies before presenting them to the crowdsourcers. We decompose privacy policies into smaller segments, and create HITS which ask questions on these policy segments. This way, we simplify the labeling procedure and minimize the worker efforts, improving the acceptance rate of the labels.

3.2 Label Reliability

One of the major challenges when using crowdsourcing techniques is to ensure data quality and reliability. Previous studies show that there should be an emphasis on quality control when dealing with crowdsourcing tasks because they may have varying degrees of skill or may not pay full attention when performing HITs [11]. Wilson et al. [54] design a system to predict and highlight important paragraphs in the privacy policies, leading to an increase in the annotation accuracy. However, this may also result in an unwanted situation where the labelers only focus on the highlighted sentences but no other texts, especially when they are crowdsourcers with no background knowledge. The authors also provide definitions for privacy-specific terms used in the questions and the response options (e.g., third parties, explicit consent, core service, etc.). Those clarifications are provided as pop-ups when the user hovers over a term highlighted in blue.

Lovett et al. [31] confirm the positive correlation between worker payment and annotation accuracy. He claims that if payments seem low for the level of effort required by the survey, the respondent may be more likely to reduce efforts when responding. He also suggests to include an "estimated time to complete" in the task descriptions.

Other research shows that simplifying wording and sentences is a good way to obtain data with better quality. The UI layout and survey questions are also important aspects to increase the annotation accuracy. Wilson et al. [54] also introduce a bonus paid in addition to the agreed compensation rate to MTurkers who do particularly well on tasks. In addition, attention checkers are necessary to ensure the crowdsourcing data quality [38, 33, 21]. We leverage similar techniques along with a majority rule based voting system in our design. Details are presented in Chapter 4.

3.3 Privacy Policy Analysis

Given the number and length of privacy policies, much of the related work focuses on extracting important information from them and identify the related data practices. While most prior works on automated privacy policy classification use statistical models or simple machine learning classifiers, and some of they study the inconsistency between app behaviors and their privacy policies, Calpric focuses on privacy policy classification solely, and is developed based on the state-of-the-art language models with a customized policy word embedding.

Watanabe et al. [52] use keyword extraction on privacy policies to identify non-compliance between mobile apps and their privacy policies. Wilson et al. [53] built and trained text classifiers using the OPP-115 Corpus, which consists of 115 website privacy policies labeled by legal experts. Using the same data corpus, Liu et al. [29] presented a performance comparison among Logistic Regression (LR), Support Vector Machine (SVM), and Convolutional Neural Network (CNN) models, and has used this to also detect non-compliance in Android applications [62]. Harkous et al. [17] developed a multi-label classifier using CNN to label policies using major categories and smaller attributes. Elisa et al. [11] presented a solution to automatically assess the completeness of a policy using more complex algorithms such as Linear Support Vector Machines (LSVM). Zimmeck et al. [61] also created a labeled mobile app-specific privacy policy corpus, APP-350. To compensate the rarity of negative annotation labels, they introduced synthetic data by manually changing positive policy texts into negative samples. Finally, Elisa et al. [11] extracted texts from 64 privacy policies, and labeled them by a single annotator. Because Calpric leverages crowdsourcing to efficiently label samples, it is able to build a larger labeled dataset and achieve significantly greater accuracy using machine learning than previous work. In addition, while Zimmeck et al. [61] used synthetic training points to address class imbalance, Calpric overcomes the imbalanced training set by mining a large unlabeled dataset for potential negative samples and gets them labeled with active learning.

Automated privacy policy classification are mostly implemented using supervised machine learning techniques, which require large annotated corpora for training and testing purposes. Most prior works trained their models on OPP-115 or APP-350, which are annotated by multiple legal experts. Although Zimmeck et al. [60] and Wilson et al. [54] also explore the crowdsourcing option, in both studies, labelers are required to read through the entire privacy policy and answer questions accordingly. In contrast, we pre-preprocess policies into segments that address the same data practice to reduce labeling effort, as suggested by Schaub et al. [41]. Whereas Wilson et al. [54] focus on methodology to increase the crowdsourcing productivity, such as highlighting the most relevant paragraphs in a privacy policy, we shift the focus and extend our work to classification of policy segments. Our study also differs from all prior published work on privacy policy classification because we integrate active learning algorithms on top of regular classifiers. Combined with the use of policy segments, Calpric is able to address the class imbalance issue by automatically querying more negative samples in the unlabeled training pool.

3.4 Active Learning

Active learning is an iterative training strategy where an initial model is trained in the normal fashion, and it is then allowed to select unlabeled training instances using a *query strategy*. In each active learning

iteration, a number of selected labels are sent to a *query oracle* for labeling and the model is then updated with the newly labeled training points—we use Amazon mTurk as our query oracle. The model can be trained for an arbitrary number of such iterations until a pre-defined stopping point is reached. In practice, the active setup is proved to achieve the same performance goal using fewer training budgets.

In general, there are three active learning scenarios: membership querying synthesis, stream-based, and pool-based selective sampling, introduced by Settles [42]. The following paragraphs summarise the three scenarios:

1. In Membership query synthesis, the learner is not only be able to select unlabeled instance in the input space, but also generate its own samples and request for a label. This is a very useful techniques if there is limited data available for training. However, because such models generate samples based on its knowledge, in our case, the machine may create gibberish text segments that contains arbitrary meanings and not human-readable, making it difficult for human annotators to label.

2. In stream-based selective sampling scenario, each instance is sampled from the actual distribution, and the learner immediately decides if it wants this sample to be labeled or discarded. The decision is made individually without considering other instances.

3. In pool-based selective sampling, as the first step, all unlabeled instances in the data pool are presented to the model. The model then selects the most informative samples and queries for them to be labeled. To determine whether an instance is "informative", we make use of a measurement score named *informative score*, which is assigned to each instance in the data pool.

Pool-based selective sampling has been the most well-studied scenario, especially for text classification [51] and information extraction [44]. The major advantage of this method is that it evaluates and ranks the entire set of unlabeled training points before selecting the next one to label [42]. We select pool-based sampling because it is applicable to Calpric's scenario as we have the entire unlabeled training set up front and it is the most effective and widely used sampling mode for text classification.

To our knowledge, this is the first paper that performs deep active learning classification on privacy policies. However, there are related studies using similar learning approaches in areas such as image analysis and classification [56, 15], and natural language processing (NLP). Zhang et al. [57] implemented active learning strategies on top of CNNs for sentiment analysis, whereas Shen et al. [46] investigated uncertainty-based active learning heuristic for sequence tagging on a newly proposed CNN-CNN-LSTM architecture. We build upon prior work and experiment with our PPWE-biLSTM model, which is developed based on the bidirectional LSTM introduced by Graves, A. and J. Schmidhuber [16]. We compare its performance with the fine-tuned BERT [13] classifiers, as both models are potential state-of-the-art designs for text classification tasks [59, 22]. In our work, we employ active learning methods to the automated policy classifier with batches of samples created by crowdsourcers, evaluate and compare different querying heuristics, and observe significant improvements from previous works. Details are presented in Chapter 4.

Chapter 4

Design And Implementation

We leverage active learning and crowdsourcing techniques to design an automated privacy policy classification tool, Calpric, to perform annotation equivalent to those done by skilled human annotators with high accuracy while minimizing the labeling cost. Figure 4.1 shows a high-level overview of our proposed system, comprising four major components: *data preparation* (Details are in Section 4.1), *crowdsourcing* (Section 4.2), and *querying* and *active learning* covered in Section 4.3.

The objective of Calpric is to extract whether a privacy policy declares collection for a particular private data category. Currently Calpric supports three private data categories: user contacts, location, and device ID. Due to the nature of privacy policies, it is possible that one policy segment is associated with multiple claims on different data practices. For instance, "We may ask you to provide your personal information , the scope of which is as follows : We may ask for your name , e-mail address , contact info, geographic location and the time and date of your visit ." covers collection on both contact and location information. To be able to accurately reflect this situation, instead of using a multi-class classification model, we design Calpric to be a combination of multiple binary classifiers, each being responsible for classification of one private data category. The binary classifiers are trained on policy segments of the according categories.

Calpric is a cyclic system that performs active learning on an unlabeled training pool, selects the most informative instances and queries for labels, and adds these labels to the training set for the privacy policy classifiers. The first stage of our design is *data preparation*, in which we download $375K$ Android app privacy policies from the Google Play Store. We filter out duplicate privacy policies, process and sanitize them, resulting in a total of $52K$ privacy policies in raw text format. To reduce crowdsourcing effort and increase label accuracy, instead of requiring labelers to read through the entire privacy policy, we extract smaller *policy segments* from the original policies, and pre-classify them into groups using basic keyword extraction heuristics. After collecting the *Unlabeled Train Set*, Calpric continues to the next stage: *querying*. At the beginning of each active learning iteration, the model selects the most representative segments $\mathbb{S}$ from the *unlabeled train set* or *unlabeled training pool*, denoted by $\mathbb{X}^U$. These segments are sent to mTurk and labeled by multiple Turkers, in order to reduce the possibility of erroneous labels. The assigned labels are post-processed into a single labeled policy segment and added to the accumulated labeled training set $\mathbb{X}^L$. The process of requesting and retrieving labeled data is handled by the query oracle. The stage in which the query oracle queries unlabeled instances to be labeled

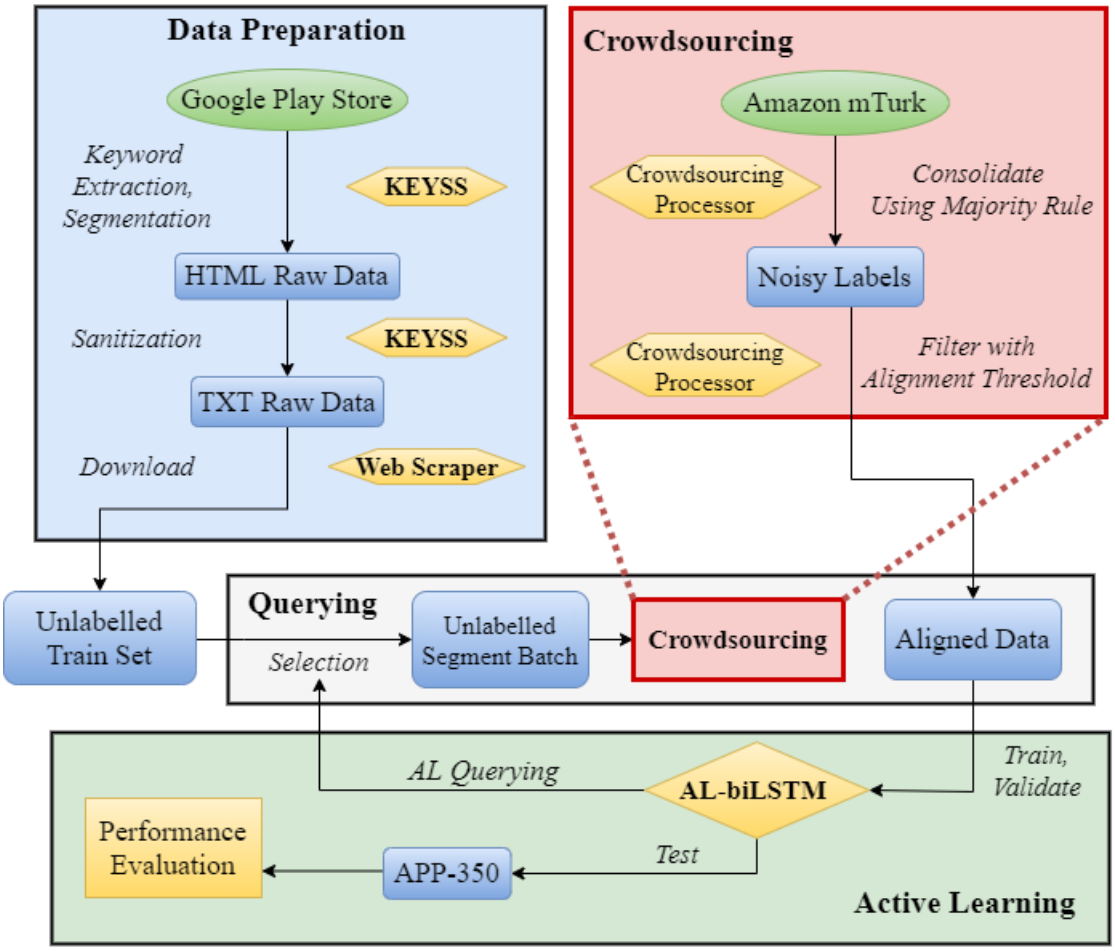

Figure 4.1: A simplified overview of the active learning system

is called *Crowdsourcing*. Moving to the last phrase, *Active Learning*, Calpric trains the classification model and updates the weights ω using the training set with newly added labels. We repeat the entire process of all stages until a pre-defined stopping criterion is satisfied, for instance, after the average f1 score reaches 70%. We provide detailed descriptions on design and implementation in the following three chapters.

4.1 Data Preparation

As an application of iterative supervised learning, Calpric requires a small set of labeled privacy policy segments to boot-strap the active learning process and a large set of unlabeled privacy policy segments. The data preparation process begins with privacy policy acquisition, which is covered in Section 4.1.1, followed by policy segmentation in Section 4.1.2.

4.1.1 Policy Downloading

We scrape Android application metadata, including the link to the app's privacy policy, from the Android Google Play Store. We then filter out broken or invalid privacy policy links. We only consider policies that are in HTML format, though we could have easily extended our preprocessing to handle less common formats such as PDF or raw text. Our web scraper is developed based on the dragnet's pre-trained model [37], which is a content extraction tool to separate the main content of a web page from the navigation chrome. Specifically, we make use of the following two models: *kohlschuetter readability weninger comments model* and *kohlschuetter readability weninger content model* [25]. The downloaded HTML is then passed through a sanitization pipeline, in which the irrelevant elements, including HTML tags, advertisement, and UI features (eg., navigation bar), are removed. We verify that the downloaded pages are actually privacy policies by checking for the presence of keywords such as "privacy policy" and "legal" as well as by excluding any documents with fewer than 50 words. Finally, we verify text language using the Python langdetect library [36] as the Calpric models currently only handle English text. Finally, we identify and remove duplicates from the dataset, leaving 51,781 usable privacy policies.

Algorithm 1 shows the pseudo codes for our web scraper:

4.1.2 Policy Segmentation

Unlike legal experts who have professional knowledge and experience on privacy policies, it is difficult for crowdsourcing workers to read through a complete policy, extract key information and label segments accordingly. In order to lighten their burden and increase the annotation accuracy, we preprocess privacy policy texts and segmented them into smaller segments before sending them to crowdsourcers to perform annotation.

Therefore, as the next step of data preparation, we extract policy segments, which will form the training points for Calpric. Policy segmentation includes the following tasks: keyword filtering, segmentation and pre-classification.

We first tokenize each privacy policy into a list of sentences using the Python NLTK library [6]. Calpric then filters the sentences using keywords corresponding to three private data categories: contacts, location and device ID. For example, we use keywords such as "email" and "phone number" to extract

Algorithm 1 Web Scrapper

Input
I applications IDs of which privacy policies to be downloaded
Output
O downloaded privacy policies in html and txt format

```
1: procedure DownloadPrivacyPolicy(I)
2:     O = ∅                                          ▷ initialize an empty output set
3:     for each ID i in I do
4:         app_data ← parse and load app info from Google Play
5:         policy_url ← extract link of privacy policy from app_data
6:         if policy_url ≠ ∅ then
7:             content ← try naive download without javascript emulation
8:             if content = ∅ then
9:                 content ← full download using webdriver.Chrome
10:            o ← sanitize content, eliminate web page features
11:        O = O ∪ o                                  ▷ save the newly downloaded privacy policy
12:    return O
```

sentences that discuss the handling of private data in the contacts category and keywords such as "IP address" for the device ID category.

Since we are using crowdsourcing techniques for label annotation, we need to pay extra attention to prevent workers from giving gibberish answers and false labels. As mentioned, to reduce their reading effort, we deconstruct the whole privacy policy into smaller segments. When doing so, we need to ensure that pieces of policy are not out of context after segmentation. At the same time, we need to minimize the segment length in order to reduce labeling effort for crowdsourcers. Recall that policy segments are smaller pieces extracted from the complete privacy policy. Since later sentences may change the meaning of earlier ones, and some sentences are meaningful only when you read and interpret them as a whole, a good policy segment needs to include all necessary sentences. On the other hand, the segment also needs to exclude irrelevant details as it places a greater burden on the crowdsourcers to read through a longer piece of text. A typical processed segment usually consists of one to three sentences, which is a reasonable length even for non experts to read and summarize with ease. In other words, the objective of segmentation is to include only the necessary sentences required to understand what the privacy policy is declaring with respect to the private data category. A typical example of a policy segment is as follows:

> Personal information is data that can be used to uniquely identify or contact a single person. When you visit, download or upgrade our app or our products, we do not use this information explicitly. However, we may collect personal information to improve our services and deliver a better experience.

All sentences in the segment should contribute to the meaning of the segment. For instance, if the segment is constructed without the first sentence, it is not clear personal information refers to contact information. The lack of descriptions and examples leads to a vague policy segment, making it difficult for crowdsourcers to provide labels. Different labels will be created based on different interpretations. While some may label these segments as a collection/use of contact information, others may not. Such ambiguous sentences are undesirable for labeling as they inherently do not have a well-defined label. On

the other hand, if the segment only contains the first two sentences, readers may be misled to provide a negative label to the segment, while in fact the app will be collecting personal information. Finally, the sentence immediately following the snippet was "Please be aware that when you register and set up an account, you will at minimum have to download the Application onto your mobile device." This sentence is not required to understand whether the application was collecting private contact information or not, and thus should not be included in the segment.

We implement a segmentation algorithm to determine where the policy segment boundaries are. Segments are constructed in two steps. First, Calpric checks for keywords such as "include" or "for example", as well as punctuation marks such as semicolons and question marks, which indicate that the previous sentence may contain information related to the current one. For instance, if the current sentence starts with punctuation such as comma, colon, semicolon, etc., we combine the previous sentence with the current one to form a segment; if there exists special punctuation marks, such as question marks, at the end of the current sentence, we combine current sentence with the next one to form a segment.

Second, Calpric uses NLP algorithms to measure the similarity of consecutive sentences. As a part of the algorithm, similar to Harkous et al. [17], we create a domain-specific word embedding trained by the 52K downloaded policies on top of Fasttext [7] and use it to generate vector representations of sentences. While there are various general-purpose pre-trained word embeddings available, customized embeddings tend to achieve better performance for classification tasks [50]. As a Fasttext-based embedding that allows vector training on sub-words, the policy embedding is able to interpret words with spelling mistakes and, more importantly, captures the actual meaning of proper nouns, which usually consist of several words.

Calpric measures sentence similarities using the Word Mover's Distance (WMD) [27], which evaluates how relevant the previous sentence is to the current one. Dias et al. [14] propose a threshold function that consists of the average and the standard deviation of the downhills depths, where downhills represent topic shifts in a document. To determine if there is a topic boundary between two sentences, our tool compares the WMD similarity of these sentences against a *Segment Threshold* (ST), which is calculated for every individual privacy policy; if the similarity is larger than the threshold, the two sentences are considered to be related. Therefore there should be no boundary between them.

ST is calculated as follows:

$$ST = \mu + Topic\_Boundary\_Constant \times \sigma, \tag{4.1}$$

where μ and σ are the WMD mean and standard deviation of sentences in a privacy policy, respectively. The topic boundary constant value in Calpric is set to 2.5.

After segmentation, Calpric now has a set of unlabeled segments labeled by private data category $\mathbb{X}^U_{contact}$, $\mathbb{X}^U_{device}$, and $\mathbb{X}^U_{location}$. We refer the entire pre-classified yet unlabeled data pool as $\mathbb{X}^U$, which is ready for crowdsourcing and active learning. Referring to the above sample segment, it contains the collection keyword *collect* and category keywords such as *contact* and *uniquely identify*. Therefore it is added to the *CONTACT* pre-classified group. Note that each segment can be added to more than one group, as it may contain texts related to collection/use of multiple user information.

In total our 52K privacy policies produced 153K, 63K, and 38K segments for contacts, location, and device ID, respectively.

4.2 Crowdsourcing

Individual segments are labeled by crowdworkers hired to perform Human Intelligence Tasks (HITs) on the Amazon mTurk service. Each segment is labeled by five workers and we consolidate the the results into a single label, which we add to the labeled training set $\mathbb{X}^L$ for the respective data category. In this section, we cover the design of the HIT survey questions in Section 4.2.1, and methods to ensure data reliability in Section 4.2.3. *The use of human workers was approved by our institutional review board (IRB).*

4.2.1 Survey Design

In previous works, when legal experts perform annotation, they need to read through an entire privacy policy, select relevant sentences and manually label them into different categories [54]. However, it is difficult for crowdsourcing workers to provide high quality annotations in this way, given the fact that the average length of privacy policies exceeds 2,500 words [35]. The labeling process introduced in previous works also limits the use of multiple workers annotating on the same segment.

In contrast, we use crowdsourcing resources for policy labeling, specifically, Amazon mTurk, which is a platform for requesters to publish HITs in the form of surveys to an open market. Crowdsourcing workers are able to search for these HITs, accept and work on the tasks. To ensure that the survey is easy to understand so the crowdsourcers are able to provide accurate labels by answering the questions, we design the survey using an iterative basis: we first develop a new version of the survey, including questions, response options, instructions and definitions. It is then reviewed by legal professionals and revised accordingly.

Because some Turkers may incorrectly label other policy segments, each survey is sent to five Turkers and we measure the level of agreement between the Turkers for each segment, and only accept labels where there is sufficient agreement. In addition, some privacy policy segments are inherently ambiguous and open to interpretation. While privacy policies were usually labeled by legal experts in previous works. We argue that such professional knowledge should not be necessary: privacy policies act as the primary mean to obtain consent from app users with no law background or legal training, but are expected to understand their content before agreeing and signing the documents. As a result, if regular people cannot agree on the meaning of a policy segment, then it has no definite meaning, and thus is not a useful training point for Calpric. We denote such segments as *ambiguous* and aim to exclude them from our labeled training set. That is, if the AR for a segment falls below a certain threshold, it will not be assigned to any label.

To ensure that our survey questions are clearly worded and cover all label cases, we experimented with several versions and narrowed it down to 4 that we felt were the clearest. To select the best one that would generate the highest quality labels, we create HITs using test segments from APP-350 that have been labeled by knowledgeable workers and publish them to mTurk using each version of the survey.

Below we show five survey versions and the questions used for the Amazon mTurk HITs. Note that the survey questions are adjusted depending on the type of private information we are asking the Turkers to label (i.e., “contact” would be adjusted to “location” and “device ID” accordingly).

Initially, we design a simple survey consists of only one question, asking whether there exists a data

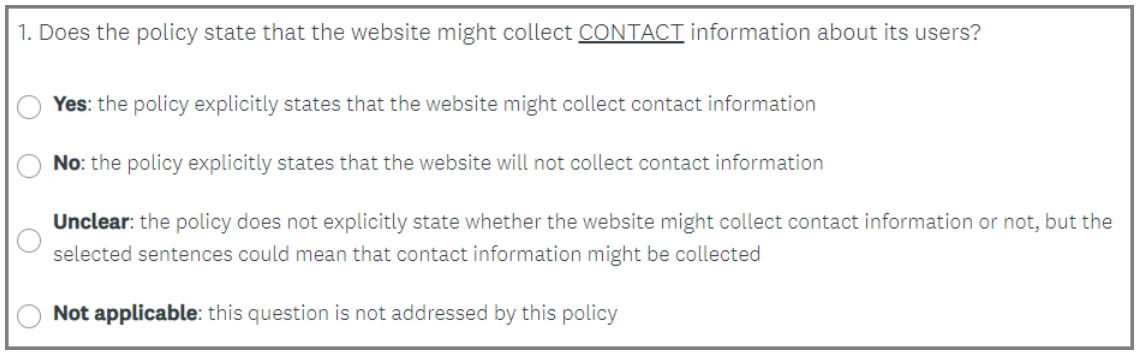

Figure 4.2: Survey version one: a simple question

collection practice associated with the provided privacy policy segment. For instance, for the segment "When you view the site using a mobile device, in addition to the other information we may collect from your actions, we may collect your mobile phone number, device identifier, and geo-location information as well as other non-personal information, such as your device type and carrier.", we ask crowdsourcers to answer the following question, as shown in Figure 4.2.

Clearly, this policy segment claims to collection contact information and the above question should be answered as *Yes*. Nevertheless, the segment not only mentions contact but also location and device information. If we proceed with this survey design, to obtain the complete label for the example segment, we need to publish three HITs for the three data categories accordingly. We can also use a combined question to reduce the number of batches needed to be published, as shown in Figure 4.3.

Instead of focusing solely on the type of data one claims to collect, the above survey is also able to differentiate first and third party data practices. Consequently, a more complex survey is needed for collecting additional information. Still, the answers to these questions are straightforward. As an example, labelers should select "1st Party Owner" for the first multiple-choice question, "contact", "device" and "location" for the second question, which is a check-box question that allows multiple options to be selected. A confirmation of data collection should be the answer to the last question.

However, after reviewing by legal experts, some revisions are made based on how we structure the questions. For instance, in this version, question one and three are related to each other. Such questions are not only confusing to labelers but also makes data processing and result memorization more difficult. We modify and update the survey as shown in Figure 4.4.

This survey not only resolves the existing issues, but is also able to capture positive and negative samples in a more straightforward way. While being very complete, it requires 4 times of labeling effort as the original survey, which consists of only one simple question. Considering the focus of this project only involves first party data practices, we simplify the survey questions to the following two versions, which we expect to be the most competitive candidates among all versions.

Version five provides very simple questions for crowdsourcers to answer. It separates data categories as Version One, which requires less effort from the labelers. Note that it shares the same disadvantage as Version One, where we need multiple HITs for different data categories. In contrast, Version Five adds a slight difficulty by combining the three data categories in one question, but saving the labeling cost by publishing one HIT for each policy segment. Figure 4.5 and 4.6 show the two versions.

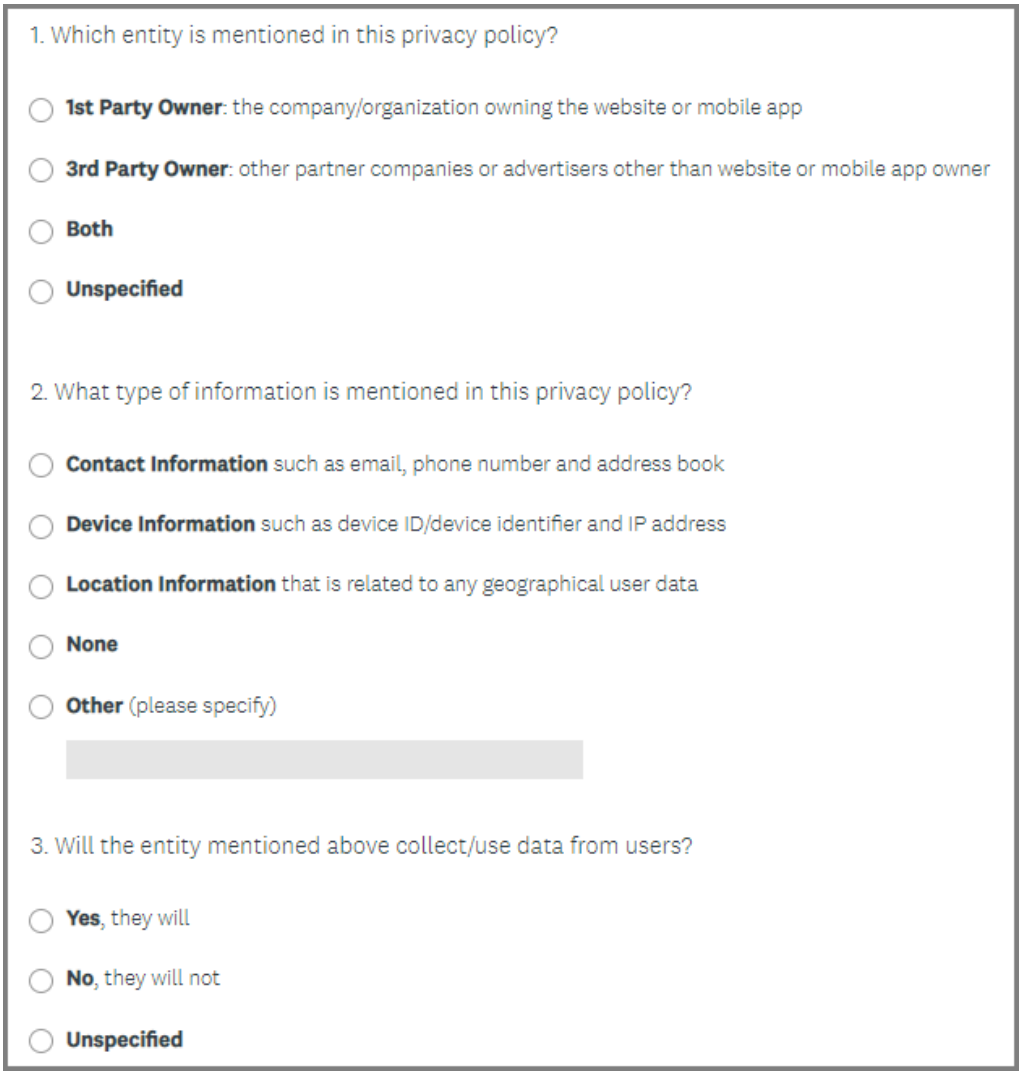

Figure 4.3: Survey version two: reduces the number of published HITs

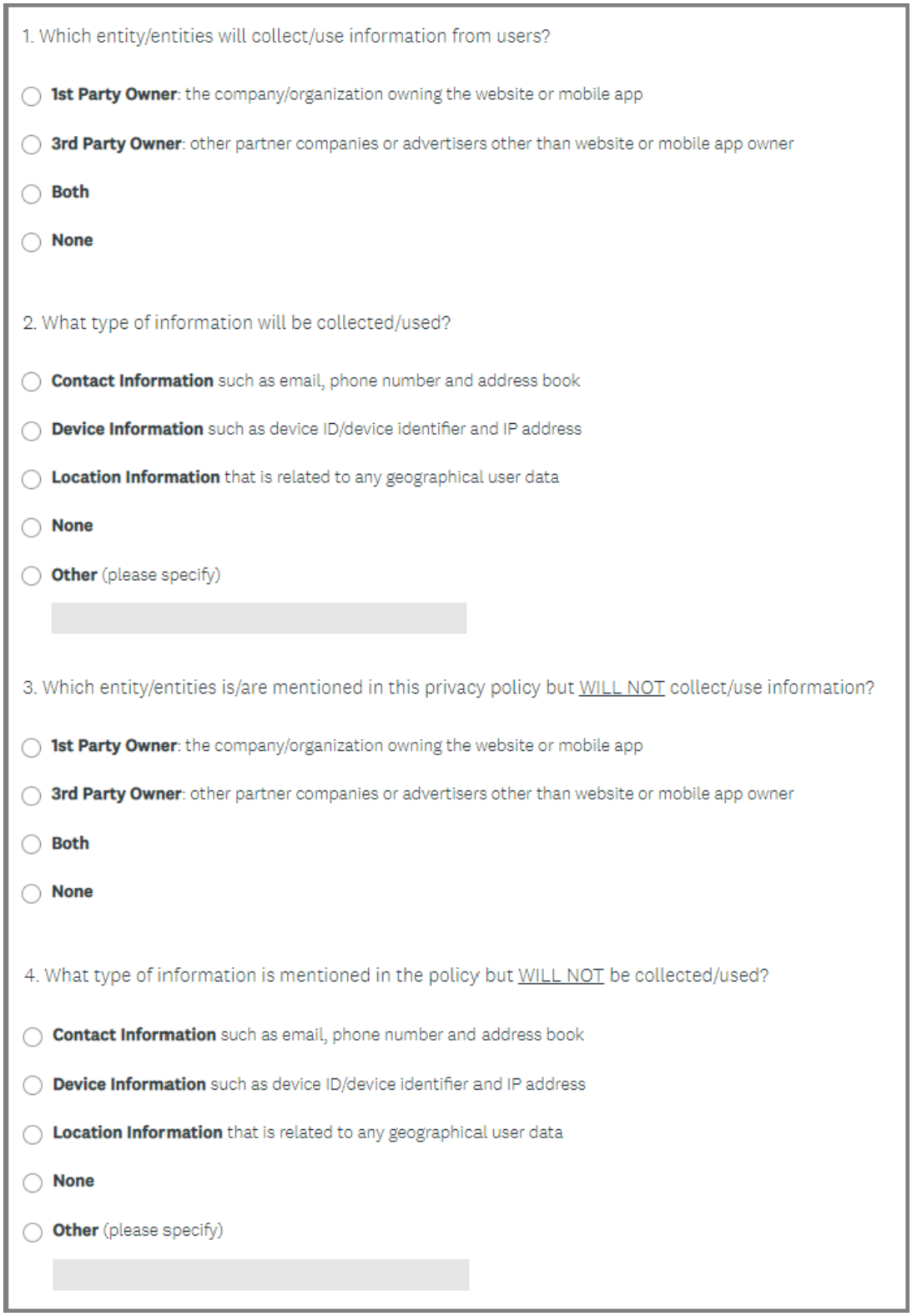

Figure 4.4: Survey version three: updates to independent questions

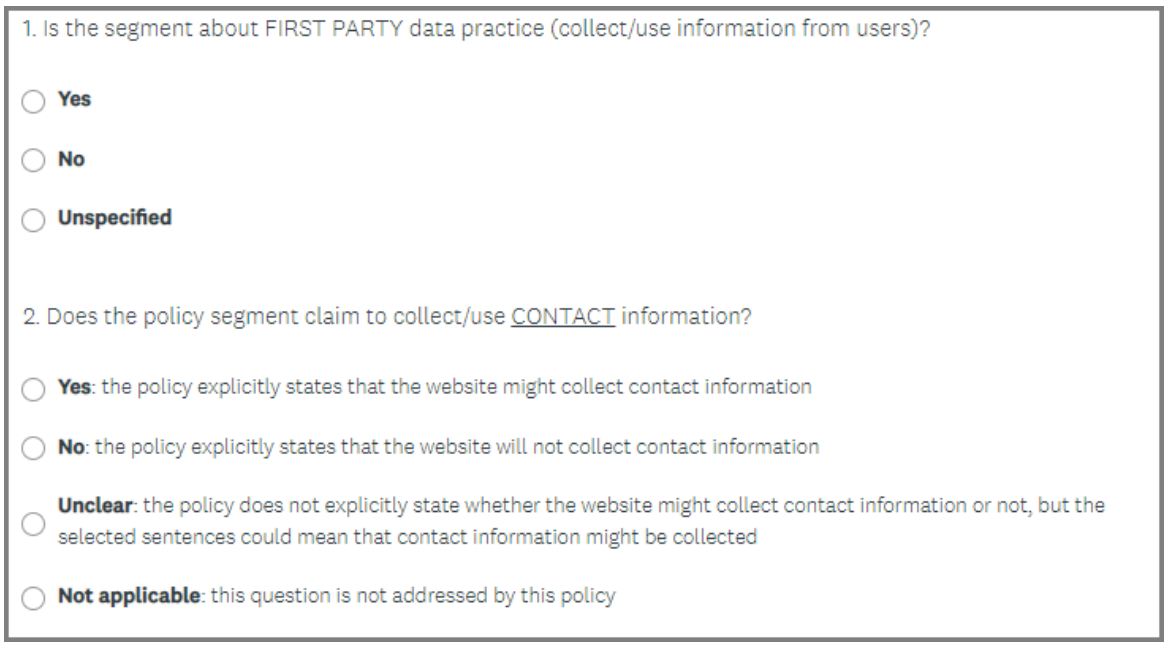

Figure 4.5: Survey version four: simplified questions dedicated for a single data category

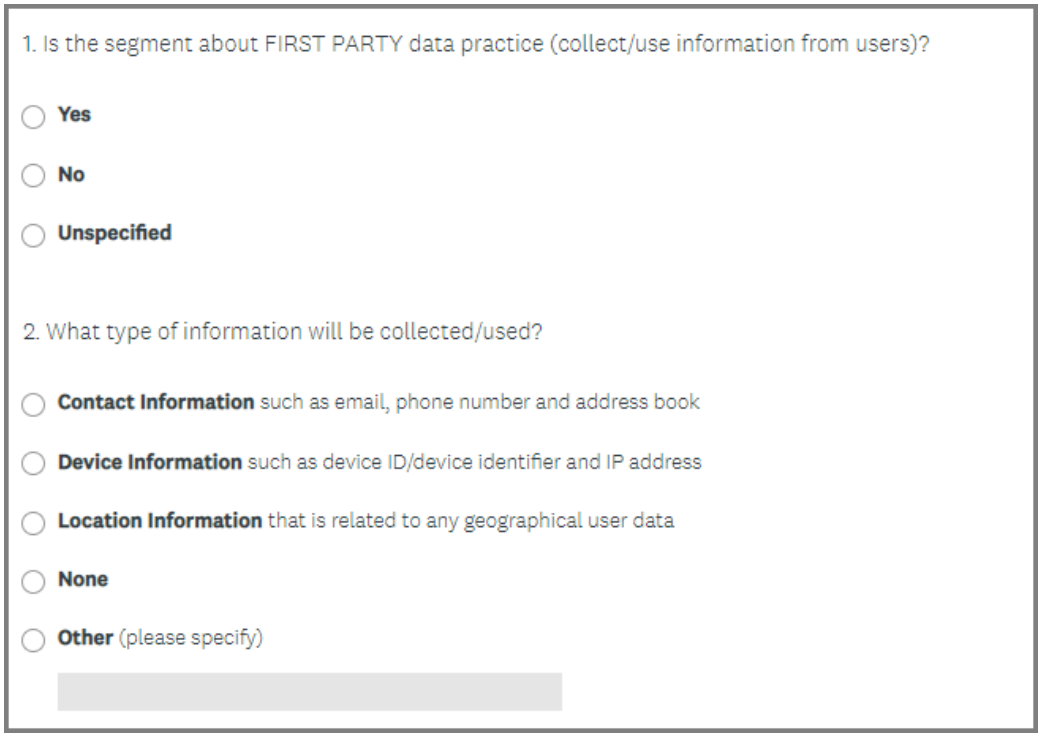

Figure 4.6: Survey version five: general questions designed for combined data categories

	# of Questions/Answers	# of Aligned Answers	# of Aligned Labels	% Aligned Answers	% Aligned Labels
Version 1	1 * 30 = 30	22	22	73.3%	73.3%
Version 2	3 * 30 = 90	41	13	45.6%	43.3%
Version 3	4 * 30 = 120	82	16	68.3%	53.3%
Version 4	2 * 30 = 60	44	22	73.3%	73.3%
Version 5	3 * 30 = 90	69	19	76.7%	63.3%

Table 4.1: AR results for different survey versions using 10 segments for each data category (a total of 30 policy segments)

4.2.2 Survey Performance Comparison

Our final survey questions were selected from the above survey variants we had constructed. To determine the best one, we computed the Alignment Rate (AR) for each survey variant. To calculate AR, we first need to introduce several definitions. Agreement Percentage (AP) refers to the percentage of Turkers labeled a segment the same way. For instance, if five workers select *Yes* for a segment, two select *No*, and yet another three others select *Other*, the AP will be $5/(3+2+5) = 50\%$. To filter out low-confidence labels, we pre-define a threshold above which the AP must fall for the label to be considered reliable (i.e., the segment is not ambiguous and the Turkers labeled the segment correctly). We call this threshold the Acceptance Threshold (AT) and default it to 80% in this study. We refer labels that pass the AT as *Aligned Labels*. AR is thus the number of aligned labels extracted from segments over the total number of segments published for labeling. From this we can see that when used on the same set of segments, and run over a sufficiently large number of Turkers, a survey variant that achieves a higher AR is likely more consistently interpreted by the Turkers and will thus yield fewer erroneous labels. We note that this simple AR metric achieves a similar result as more complex measures of correlation of internal consistency as our labels are binary (collect or no-collect).

We measure AR for our variants by preparing three batches of questions, with 10 different policy segments per batch. We randomly select the segments from the APP-350 dataset, which were labeled by skilled labelers and we consider them as 100% reliable. Using four different versions of survey questions, we calculate the AR on 30 policy segments drawn from the APP-350 dataset, with each segment labeled by five Turkers. The AR varies from 45% to 73%, with the average labeling accuracy being similar for all versions. We select the version with the highest AR for all future crowdsourcing tasks. Note that when calculating AR for this experiment, as long as there exists one or more agreed options in the check-box multi-selection question, we consider this as an aligned answer. That is, for a segment that is pre-classified as *contact* related, if all labelers agree on the collection of such information (which is associated with the third question) while two of them believe the segment also claims to use *location* information, we consider the AR of this label to be 100% because the *contact* label is aligned. The finalized annotation will then be a positive sample for First Party collection/use of *contact* information.

According to the experimental results shown in Table 4.1, the best performance occurs when using survey version 4. Note the difference between *percentage of aligned answers* and *percentage of aligned labels*: the former is calculated by the number of agreed answers divided by the total number of answers generated from crowdsourcers, whereas the latter is the number of consolidated labels divided by the number of policy segments published to be labeled. As each survey contains different number of questions,

the total number of questions and answers vary one to another.

As mentioned previously, survey version 2 suffers from a low AR because the questions of this survey are internally related. That is, if the crowdsourcers disagree on the first question, there is a high possibility that they will not reach to an agreement for the last question. We also observe a relatively large decrease in the *percentage of aligned labels* compared to the *percentage of aligned answers* in survey version 3 and 5, as these surveys consist of more questions that others. As the number of questions dedicated for each segment increases, the complexity and difficulty of the survey increase. when testing on these two surveys, for each policy segment, the 5 crowdsourcers are likely to reach an agreement in most but not all questions. As an example of survey 3, if all crowdsourcers agree on the first three questions but not the last, despite having 66.7% as its *percentage of aligned answers*, the policy segment can not be considered as *aligned* and will not result in a usable label. Since only a perfectly agreed segment can be seen as aligned, the *percentage of aligned labels* for this segment is 0%, which is significantly lower than the average *percentage of aligned answers*.

While survey version 1 shares the same AR performance as version 4, the former survey covers less information and is considered as incomplete since it does not differentiate first and third party data practices. We therefore consider survey version 4 as the final chosen version of our Amazon mTurk HITs. To summarize, it consist of three questions: the first question confirms that the policy segment is a *First Party Collection/Use*, while the second question asks whether or not the policy segment claims to collect/use private data of that type. Labels for segments where Turkers answer "No" for the first question are discarded, as these segments are likely not relevant to data collection. For the remaining labels, the answer to the second question is converted to a *positive label* if Turkers answer "Yes" and a *negative label* if Turkers answer "No". The third question collect additional information, which can be used to determine if the segment is a negative segment or not.

4.2.3 Label Reliability

Because we create training labels by crowdsourcing instead of legal experts, we need to ensure label reliability and reduce data noise as much as possible. Previous studies show that there should be an emphasis on quality control when dealing with crowdsourcing tasks because they may have varying degrees of skill or may not pay full attention when performing HITs [11]. We introduced a set of quality controls to validate crowdsourcing workers and filter out low-quality data.

Worker Requirements

The published HITs only allow qualified workers to answer survey questions. We set the following requirements on Amazon mTurk:

- Approval Rate > 85
- Number of HITs Approved > 50
- English Speaker
- Android Mobile User

Consolidating by the Rule of Majority

As mentioned, the policy texts were processed and segmented into small segments, and labeled by crowdsourcing workers on Amazon mTurk. After gathering segment labels from the workers, we consolidate the results using the rule of majority: most of the annotators must agree on the same label. If the AP of a certain label is smaller than the pre-defined AT, we consider it as a low confidence label and do not include it in our training set, as it may contaminate the classifiers. Alogirthm 2 shows the pseudo codes for our consolidation algorithm to identify whether a certain type of information will be collected or not.

Algorithm 2 Consolidate Raw Turked Results into Segment Labels

Input

S	unlabeled policy segments published to MTurk
R	raw result files downloaded from MTurk
v_{AT}	pre-set Alignment Threshold value

Output

L	consolidated labels for all segments

procedure ConsolidateTurkedLabels(R, v_{AT})
  while $R \neq \varnothing$ **do** ▷ loop through all batches of results
    $r \leftarrow$ *current result file*
    for results of each segment r_s in r **do**
      $s \leftarrow$ *current segment*
      for results of each question r_q in r_s **do**
        $r_{q+} \leftarrow count(r_q == yes)$
        $r_{q-} \leftarrow count(r_q == no)$
        $majority \leftarrow max(r_{q+}, r_{q-})$
        $v_{AP} = majority/(counts(r_q))$
        if $v_{AP} > v_{AT}$ **then**
          $L_q \leftarrow majority$
        else
          mark L_q as ambiguous
      $L_s \leftarrow \sum^{Q_s} L_q$
    $R = R \setminus r$ ▷ remove the processed result from the to-do set
  return $\sum^{S}(s, L_s)$

When consolidating labels, we loop through all privacy policy segments and process the results one by one. For each segment, we look at answers generated by the five crowdsourcers and compare their values. We use the rule of majority and AT to find the aligned labels, separate positive and negative samples and save them alongside the policy segment texts. Note that we also include a label for *ambiguous* segments, which are segments that do not have an agreed label.

Repeated Workers

Due to the nature of majority rule, we need to ensure that no worker answers the same question repeatedly. We implement a qualification test embedded in every published batch to check the worker IDs that have previously accepted the same batch of questions. Permission to access our survey is only given to Turkers who had not participated in the same question.

Specifically, the repeated worker check is implemented using the mTurk developer APIs [1]. We main-

[1] `https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/ApiReference_OperationsArticle.html`

tain a list of $WorkerId$ and check whether the incoming request is generated from a $WorkerId$ that is already in our history list of workers who performed the same task. We use the $RejectQualificationRequest$ operation to deny past participants from accessing one HIT repeatedly. We also provide a text message explaining that the rejection is to ensure our data quality, so worker who made the request can read this message.

Knowledge Test and Honesty Test

There are cases where crowdsourcers provide bogus answers without actually doing work [54] or where adversarial bots that enter spam answers on crowdsourcing platforms to earn money [34]. To prevent noise from such interference, we aim to design a qualification test in the form of simple questions to verify if the workers are qualified to answer our questions.

We first consider a knowledge test, which uses segments from the APP-350 as a test, combined with other unlabeled segments to see if the Turker is able to label the test segment correctly. If the Turker fails the test, we do not use any of that Turker's labels. To test the effectiveness of the knowledge test, we randomly select 60 questions from APP-350 and experiment on two surveys, randomly designating a question in one survey as the knowledge test and having no knowledge test question in the other survey. Each survey is given to five Turkers to perform. Our results show that many Turkers who answered the test question incorrectly still provide useful labels, and the difference in accuracy between the survey with the knowledge test and the survey without is 100% and 97.6%, respectively. For this small difference in accuracy, we would have had to discard a large number of surveys—using the knowledge test we would need to run twice as many surveys as without to achieve the same number of labeled segments. We thus conclude that such knowledge was not effective in improving label quality.

We evaluate the performance using two metrics: the label accuracy and the Label Turnover (LT), which is the ratio of *number of published segments* to *number of correctly labeled segments*. The LT ratio is a measurement of how efficiently the labeled data can be collected from the crowdsourcing workers. The results in Table 4.2 indicate that, for surveys with a knowledge test, we need to publish 3 times of the number of labels we expect to collect, whereas surveys without such tests only need 1.46 times of the expected number. That is, the data collection process is greatly delayed if we include the knowledge test.

After manual inspection of the labels, we observe that some workers label all segments correctly but did not the knowledge test question. Under such circumstances, all his labels are discarded even though they are in fact correct ones. This leads to a low LT ratio for surveys with the added knowledge test, as many labels are filtered out by mistakes. We conclude that the proposed test is inefficient and ineffective. On the other hand, we also observe that there are a few irresponsible Turkers who labeled segments by randomly selecting options instead of actually doing the questions.

To address this issue, instead of using the knowledge test, we implement a simple honesty checking question, where we ask the Turkers whether they paid close attention to the questions and provide answers accordingly, and highlight the fact that they will still receive full payments even if they did not. Such questions were proved to be effective in improving labeling accuracy [38]. We performed a similar test survey with the honesty checking question and discarded all labels of Turkers who answered "No" to the honesty checking question. The accuracy of this survey was 97.7% and when deployed, we found

that the honesty question was answered "Yes" 99.63% of the time. We include the comtesting results in Table 4.2.

	Aligned Labels	AR	Correct Labels	Accuracy
With Knowledge_Test	20	33.3%	20	100%
Without Any Test	42	70.0%	41	97.6%
With Honesty_Test	43	71.7%	42	97.7%

Table 4.2: Impact of knowledge test and honesty checking question on data collecting performance using 60 test segments

Turker Wages and Performance

We experimented on worker wages and decided on a payment of $0.60 per batch per Turker. The average time to complete a 40-question survey is 7 minutes. As Lovett et al. [31] suggest, a standard of $0.15 per minute should be adequate for most Turkers. We tested on three different payments ($0.10, $0.60, and $1.50 per batch) using five different batches, the resulting average AR are 31.9%, 77.0%, and 76.0%, respectively, as shown in Table 4.3. Note that the $0.10 payment results in a significant drop in the AR. One batch was not even completed as no more workers were willing to take the task for such a low payment.

	Batch 1	Batch 2	Batch 3	Batch 4	Batch 5	Overall
$0.1	22.5%	12.5%	20.0%	72.5%	-	31.9% [2]
$0.6	80.0%	57.5%	82.5%	85.0%	80.0%	77.0%
$1.5	75.0%	55.0%	87.5%	85.0%	77.5%	76.0%
Avg	59.2%	41.7%	63.3%	80.8%	78.8%	63.8%

Table 4.3: Performance on alignment rate for different payments

We observe that experienced workers are not attracted by the extremely low payment, while workers who accepted these tasks did not spend much effort doing questions, as reflected by the result of $0.10. On the other hand, surprisingly, the higher pay rate of $1.50 did not necessarily produce better results. Despite that these Turkers spending more time on the labeling tasks, the overall AR did not vary much from that of the $0.60 pay rate. We observe a performance ceiling due to the fact that some policy segments are vague by nature. It is difficult to label them, even for legal experts. We therefore concluded that $0.60 per batch per worker is an appropriate payment, and use this setting in all other experiments and evaluations.

4.3 Automated Privacy Policy Classification

In this section, we present our automated privacy policy analysis tool. We discuss the basic machine learning (ML) model selection in Section 4.3.1, and active learning strategies to improve label efficiency in Section 4.3.2.

[2] Calculated based on n=4 without considering the uncompleted batch

4.3.1 Classification Model Description

Because crowdsourcing holds the promise of providing more labeled data than previous studies, we are able to use more complex, higher capacity models than previous works that which used feature selection and non-deep models such as LR and SVM. Traditional CNNs experience the issue of long-distance dependency in text classification, while recent NLP studies solve this problem by introducing RNNs, LSTM, and BERT models. These models have been proven to achieve a better overall performance in text classification [30, 59, 13].

A commonly used, state-of-the-art language model is BERT, which can be fine-tuned for various tasks including text classification. However, because active learning is an iterative training algorithm, an important factor is the retraining time of the model that is selected. While accurate, BERT suffers from slow training due to its complexity and the number of parameters involved, despite it being an application of transfer learning. Another popular classification model is bidirectional Long short-term memory (biLSTM), which is one of the most effective networks for NLP tasks but has not yet been introduced to privacy policy text classification. As a comparison between the two, Joselson and Hallen [22] evaluate a fine-tuned BERT model and a customized biLSTM classifier for sentiment analysis, and show that the former model requires hours more training while achieving similar performance (69% vs. 71%). Based on their results, we decide to develop our own model using biLSTM layers, alongside with a customized privacy policy word embedding trained on top of the general-purpose Fasttext embedding. We name our model *PPWE-biLSTM*.

As mentioned, our LSTM nodes are bidirectional, so two recurrent layers in opposite directions serve as a platform for training in the past and future of a specific time frame. The PPWE-biLSTM used in our experiments consists of a hidden layer with 100 densely connected memory units, as the structure is proved to achieve a lower perplexity than regular stacked models [23, 18]. We use leaky ReLUs [32] and dropouts [47] of 0.1 to prevent over-fitting while sustaining the weight updates to avoid vanishing gradient problems in the propagation process [32]. We apply a sigmoid activation function to the model output for binary prediction. We use binary cross entropy loss and ADAM optimizer [24] to find model weights. We used $BATCH\_SIZE$ of 20 and $EPOCHS = 4$. We apply the same vectorization method with our customized word embedding for the ML component. An evaluation of PPWE-biLSTM vs. BERT is included in Section 5.3. We also include a model system using our customized word embedding and LR, serving as a baseline comparison. Different from previous work where they generate vector representations of the segments by taking the union of a TF-IDF vector and a vector of manually crafted features [29], our classification process is solely based upon the deep learning model. For the baseline LR model, we use the default implementation of logistic regression with L2 regularizer and 0 random state in scikit-learn. Other learning parameters are set to the same as what we have for bi-LSTM. We use the above model setup as default for all experiments, unless specified otherwise.

4.3.2 Applying Active Learning

Recall that we decide to investigate pool-based sampling because we have the entire unlabeled training set available and it is the most effective and widely used sampling mode for text classification. The detailed setup is described in the following sections, including querying strategies (Section 4.3.2), alignment threshold (Section 4.3.2), and re-labeling strategies (Section 4.3.2).

Querying Strategies

As mentioned, the active learner proactively selects a subset of available examples in each learning iteration. The key question is, how does the learner identify which labels are considered to be the most *informative*? While Settles [42] describes a wide range of existing querying strategies, we first investigate *uncertainty-based sampling* [28] as it is the most commonly used query framework. In this framework, unlabeled samples are ranked according to how much confidence in predictions made by the current model. Within uncertainty-based sampling, we employ three different implementations: *Least Confidence*, *Margin Sampling* and *Entropy-based Sampling*. We further explore other querying strategies based on the *reduction* [39], the *density-weighted* [43], and the *batch mode* [8] framework:

Least Confidence (LC): The Least Confidence (LC) strategy was first proposed by Culotta and McCallum [12]. LC allows the active learner to select the least certain instance from the unlabelled set and request for it to be labelled. The confidence probability is calculated using:

$$\underset{y \in Y}{max}[1 - \mathbb{P}(y \in Y|\mathbf{x})] \tag{4.2}$$

Margin/Entropy Sampling: Other popular uncertainty querying strategies include *Margin Sampling* and *Entropy Sampling*. The former selects instances where the difference between the first most likely and second most likely classes are the smallest [40], whereas the latter chooses samples with the largest entropy in class probabilities [45]. In other words, instead of minimizing the classification error, entropy method tries to minimize the log loss. For binary classification, **MS** and **ES** reduce to **LC**. In such cases, the three methods will query instances with a class posterior closest to 0.5, that is, the most *ambiguous* segments [42]. We confirmed this conclusion using our CPPS dataset. In future evaluation, we refer to these algorithms as **LC**.

Expected Error Reduction (EER): Instead of considering individual instances along, there are methods take the entire input space into account, which have the potential to prevent sub-optimal queries. The Expected Error Reduction (EER) sampling method selects instances that make the most impact on the current model. Specifically, it measures how much the generalization error is likely to be reduced as the new samples are introduced to the model. We built upon Roy and McCallum's work [40], calculate the expected future error of a classifier using the Monte Carlo Estimation. Below shows the formula to estimate the binary loss, which is used in our evaluation:

$$\mathbb{E}_{P^*_{\mathbb{X}^L}}(binary) = \frac{1}{|\mathbb{X}^U|} \sum_{x \in \mathbb{X}^U} (1 - \underset{y \in Y}{max} P^*_{\mathbb{X}^L}(y|x)) \tag{4.3}$$

where $\mathbb{X}^U$ is the unlabelled training pool and $\overset{*}{\mathbb{X}}{}^L$ is the current training set with the chosen query (x^*, y^*) added to the original labelled training set $\mathbb{X}^L$. We calculate expected error for each possible label, $y \in \{y_0, y_1\}$ using the learner's prediction distribution $P^*_{\mathbb{X}^L}$. As the true label for x^* is unknown before the query, we use the current model to estimate the true label probabilities. We design our model with a *p_subsample* of 0.5 to improve runtime for large sample pools.

Information Density (ID): Settles and Craven [43] propose the Information Density (ID) framework. When querying new samples, ID not only considers *uncertain* instances, but also those which are *representative* of the dense region of the input space. The ID value of an instance x can be calculated

as follow:

$$\mathbf{ID}(x) = \frac{1}{|\mathbb{X}^U|} \sum_{x=1}^{\mathbb{X}^U} sim(x, \bar{x}) \tag{4.4}$$

where $sim(x, \bar{x})$ is a similarity function such as cosine similarity or Euclidean similarity. The value is calculated using x and the average similarity (denoted by $\bar{x}$) of all other instances in the input distribution. A higher ID value represents a closer relationship between the given instance and the rest of samples in $\mathbb{X}^U$.

Batch Mode Uncertainty (BMU): Traditional AL query strategies suffer from sub-optimal record selection when passing $n\_instances > 1$, that is, querying multiple instances in each iteration. The Batch Mode Uncertainty (BMU) sampling method addresses this issue by enforcing a importance ranking system for records among the batch. Our design makes use of the *modAL* active learning framework, which is built upon the study of Cardoso et al. [9]. They implement a BMU framework to prioritize diversity on the initial iterations, providing a global view of the input distribution. As the number of labelled instances increases, the system then shifts the priority to instances in which the classifier is uncertain about. The ranking score of each instance x is calculated as:

$$BMU(x) = \alpha(1 - sim(x, \mathbb{X}^L) + (1 - \alpha)LC(x) \tag{4.5}$$

where $\alpha = \frac{|\mathbb{X}^U|}{|\mathbb{X}^U + \mathbb{X}^L|}$, $LC(x)$ is the uncertainty of predictions for x calculated by the LC algorithm. The similarity function $sim(x, \mathbb{X}^L)$ measures to what extend the feature space is explored near x. The BM score is calculated for all x in $\mathbb{X}^U$, and ranked in ascending order. In each AL iteration, a preset number of instances are selected from the pool in this order, and the BM scores will be re-calculated.

Acceptance Threshold & Alignment Rate

As described in the previous section, the acceptance threshold (AT) defines the minimum agreement percentage (AP) among labelers that must be achieved for a label to be accepted. There are several prior studies on privacy policy classification (without AL) using crowdsourcing methods, in which 80% and 100% are the most common thresholds [49, 54].

Setting the AT has a clear trade-off: increasing the AT will result in a decrease in Alignment Rate (AR), which eventually leads to the query oracle needing to create more mTurk HITs to achieve the same number of labeled samples. This directly increases the cost of labeling for Calpric, but may lead to higher quality training data. However, we also need to take into account that some segments may be inherently ambiguous and thus may never achieve an AP greater than the AT, regardless of how many HITs are published.

As a result, while Calpric uses a default AT of 80%, it's AT is configurable and we evaluate the effect of AT on Calpric's accuracy in Section 5.5.

Re-labeling Strategies

Recall that labels for segments where the query oracle doesn't achieve an AP above the AT are not added to $\mathbb{X}^L$. Calpric implements two options for what will be done with such segments:

Label and Discard: This option compensates for the possibility that some number of labels will be

discarded as they do not pass the AT. As a result, in order to achieve the 30 labels Calpric aims to add to $\mathbb{X}^L$ in each iteration, we use the average AR to estimate the total number of segments we should submit for labeling. In our experiments, we estimate AR for our survey to be 73%, which requires Calpric to request 42 segments to be labeled on each iteration. Segments that do not meet the AT are discarded and can never be selected by the query strategy again based on the assumption that they are inherently ambiguous. Note that using this approach, the amount of new labels introduced in each active learning iteration may not be consistent.

Incremental Re-labeling: The above strategy may be overly conservative in discarding segments that fail the AT test after only one iteration, as segments may also fail to pass the test due to poor crowdsourced labelers. Discarding such segments reduces the overall unlabeled pool the querying strategy can select from. This motivates an incremental re-labeling strategy, which we implement based on that of Zhao et al. [58]. In this approach, we publish the exact number of segments we aim to be labeled (30). Instead of discarding segments that fail the AT test, Calpric republishes those segments in following iterations. We use N to represents the pre-set number of labeling iterations (i.e., the maximum tries of labeling request on one policy segment). In our case, we set it to 3. The following example shows the workflow: in the first labeling iteration ($N = 1$), we publish 30 unlabeled segments to Amazon mTurk, each labeled by 5 crowdsourcers. For any labels that have their $AP < AT$, they are considered as unaligned and the query oracle will request an additional 5 labels for the same segment in the second labeling iteration ($N = 2$), resulting in a total number of 10 labels. We repeat the same consolidation process to check for each segment whether these labels reach an agreement. If a clear majority fails to emerge after a total of $N = 3$ tries, resulting in a total number of 15 labels on each segment, we mark the sample as ambiguous and remove it from the training pool.

Chapter 5

Evaluation

For evaluation, we present measurements on the data distribution in Section 5.1, data similarity in Section 5.2, and a comparison of PPWE-biLSTM with BERT in Section 5.3. We then evaluate the effectiveness of Calpric as compared against the the non-AL baseline model (Section 5.4), as well as it sensitivity to various configuration parameters by comparing the performance of various querying strategies. We present an evaluation on different AT values in Section 5.5, and explore different options for dealing with non-alignment in Section 5.6. Lastly, we show how the active learning algorithm solves the issue of class imbalance (Section 5.7).

5.1 Data Distribution

As mentioned, we focus on policy segments for *First Party Collection/Use*, with data types being geographical location, device information, and contact information. We extracted these labels from APP-350 and OPP-115, and grouped them into the same categories. We analyzed the two corpora and summarized the results in Tables 5.1 and 5.2. The numbers 115 and 350 in OPP-115 and APP-350 indicate the respective numbers of privacy policies covered by each dataset. The number of policies covered in the corpus is different from the number of segments with useful labels.

	Location	Device	Contact	Total
First Party Actions	403	608	1019	2030
- Positive Samples	396	604	990	1990
- Negative Samples	7	4	29	40

Table 5.1: Data distribution of OPP-115 corpus (number of segments per category)

	Location	Device	Contact	Total
First Party Actions	852	1633	1608	4093
- Positive Samples	669	1394	1270	3333
- Negative Samples	183	239	338	760

Table 5.2: Data distribution of APP-350 corpus (number of segments per category)

We observe that the number of segments of interest for APP-350 is much larger than that of OPP-

	OPP-115 & APP-350	OPP-115 & CPPS	APP-350 & CPPS	Category Average
Contact	0.79/0.77	0.83/0.75	0.82/0.76	0.81/0.76
Device	0.74/0.71	0.76/0.73	0.71/0.70	0.73/0.71
Location	0.75/0.77	0.81/0.77	0.78/0.76	0.78/0.77
Corpus Average	0.76/0.75	0.80/0.75	0.76/0.74	0.78/0.75

Table 5.3: Inter-similarity/re-segmented inter-similarity

115. Note that the web-based OPP-115 dataset is more fine-grained and covers a larger set of categories, whereas the mobile application policies in APP-350 tend to be simpler. Policy segments in OPP-115 are mostly short paragraphs while APP-350 has a smaller average length of words per segment, usually one to two sentences. Similar to APP-350, our data are collected from Android mobile applications, and our segmentation algorithm generates policy segments with their length similar to that of APP-350.

One of the existing issues in the classification of privacy policies is the lack of negative samples. As we can see from the data distribution of OPP-115, the positive/negative ratio is highly imbalanced. In fact, the average Negative Sample Ratio (NSR) is only 2.0%. Research shows that such training data in many machine learning models can result in poor performance [26]. While APP-350 addresses the class imbalance issue by introducing synthetic negative data with manual modifications, which may not be the ideal solution, its NSR is only 18.6%. In our proposed model, the active learner is able to select the most representative segments and balance the training set automatically.

5.2 Data Similarity

We conduct data similarity tests on OPP-115, APP-350 and our Crowdsourcing Privacy Policy Segments (CPPS) for two reasons: to evaluate our proposed segmentation algorithm and to show that CPPS is similar enough to APP-350 that it is feasibly to use labels in the APP-350 to evaluate our model trained in CPPS.

We extract segments from the three datasets, and select 100 in each category: contact, location, and device. We evaluate similarity on segments rather than the entire privacy policies because our classifiers are trained on policy segments. Note that even within a set of data obtained from a single source, the individual policies may be very different from one another in terms of the way they are structured, the length of the policies, the complexity, and the wording preference. To address this, we introduce three similarity measurements:

1. *Intra-comparison (Intra):* Evaluate data similarity within the same corpus, compare similarity of segments that share the same labeling categories.

2. *Inter-comparison (Inter):* Evaluate data similarity across different corpora on the same categories. (Eg., location labels in APP-350 vs. location labels in CPPS)

3. *Inter-comparison on re-segmented policies (Re-inter):* Instead of using the original policy segments in OPP-115 and APP-350, which were extracted manually by human lablers, we re-segment the

complete policy texts of these two datasets using Calpric's segmentation algorithm described in Section 4.1.2, and evaluate inter-similarity on these segments.

While many existing similarity measurements are intolerant of disjoint distributions, WMD is able to produce a smooth measure even if two sentences do not share any word in common. We therefore select WMD and use it on top of our customized Fasttext word embeddings and apply it to the 100 segments from each dataset. The overall intra-similarity for a corpus is defined as the average of all text similarity measured across the selected segments. The similarity value is represents the distance between sentences. In other words, a lower distance value indicates a higher similarity. We denote this as *Similarity Distance* (SD).

To evaluate our segmentation tool, we tabulate *Inter*/*Re-inter* in Table 5.3 by the respective values in each cell separated by slashes. *Re-inter* similarity distance are generally lower than *Inter* similarity distances due to the original segments in the OPP-115 and APP-350 being created differently. However, the average difference between *Inter* and *Re-inter* is reasonably small, only 0.3%. We therefore conclude that our automatic segmentation method produces similar segments as the ones manually created by skilled human labelers.

Our following experiments use a testing set of segments drawn from the APP-350 dataset. To ensure that the underlying distribution of these datasets are similar, we compare *Inter* and *Intra* across all three datasets. The three corpora share similar *Intra*, resulting in an average of 0.75. Compared to this value, *Inter* across different corpora indeed has a slightly higher value, with an average SD of 0.78, indicating a slightly larger difference across different corpora than within the same set. On closer inspection, we observe that APP-350 and CPPS are very similar in terms of segment structure and labels (0.77), whereas OPP-115 and CPPS is the most distinct combination among all (0.80). We therefore conclude that it is feasible to use APP-350 as the validation set, since the inter-similarity between APP-350 and CPPS is relatively close to the intra-similarity of CPPS.

5.3 Classification Model Evaluation

In this section, we demonstrate that the proposed PPWE-biLSTM gives an accuracy similar to the state-of-the-art BERT model, while significantly outperforming the baseline LR classifier, which was widely used in previous privacy policy studies. Note that our LR model does not use tf-idf based feature selection but a customized privacy policy word embedding.

We randomly selected 450 segments from each category of the unlabeled training pool $\mathbb{X}^U$. The segments were labeled by the crowdsourcing workers. We applied our post-processing algorithms to discard low confidence labels, and consolidated them into three sets of training data for contact, location, and device, each containing 300 labeled segments. We prepared three test sets, each containing 300 labeled segments randomly selected from the APP-350 Corpus.

For each classification algorithm, we trained three classifiers and evaluate them using the prepared test sets. The experimental results are shown in Table 5.4. We use *biLSTM* as an abbreviation for *PPWE-biLSTM*. We use two metrics to measure accuracy: F1 score and Matthew Correlation Coefficient (MCC). MCC, which is equivalent to the mean square contingency coefficient, represents the correlation between target and predictions. Recall that we cover the introduction to MCC and include its formula

in Section 2.5. In binary classification, MCC is more informative since it takes into account the balance ratios of the four confusion matrix categories, especially when class imbalance issues [10] exist. The value varies between -1 and $+1$, where 1 is a perfect agreement between the actual and predicted labels, -1 is a perfect disagreement, and 0 occurs when the predictions are random with respect to the actuals. We include this additional metric since accuracy and F1 scores are asymmetric and sensitive to data imbalance. We perform an additional experiment using the OPP-115 and APP-350 Corpora. Similar to the previous experiment, the classifiers are trained on 300 labels selected from each of the three categories in the datasets. As an example, Table 5.5 shows the results for the location classifiers. Evaluation results for the other two categories are included in Tables 5.6 and Table 5.7. We use *biLSTM* as an abbreviation for *PPWE-biLSTM*.

From the above tables we observe:

1. Both PPWE-biLSTM and BERT outperform the LR baseline model significantly, with an average F1 score of 89.7% and 90.1% versus 67.4%, and an average MCC value of 61.3% and 63.8% versus 10.7%.

2. Based on the F1 score and MCC values, our PPWE-biLSTM model achieves an accuracy similar to BERT, whereas its training time is significantly shorter than that of BERT. We therefore proceed to investigate active learning approaches with the PPWE-biLSTM model.

3. The average *training* accuracy, F1 and MCC of PPWE-biLSTM are $98.2 \pm 1.2\%$, $98.4 \pm 0.9\%$, and $97.1 \pm 1.6\%$ respectively. Compared with the *testing* results, there is a reasonable difference of $< 10\%$. As an additional example, Figure 5.1 shows the trend of F1 and MCC values for contact classifiers trained on 2000 labeled segments, further confirms that these setups do not suffer from over-fitting.

4. Although the classifiers trained on OPP-115 have very high accuracy, precision and recall values, the MCC accuracy is inferior due to the unbalanced dataset. The lack of negative samples results in nearly no training on *DoesNot* data actions. That is, the classifier predicts every label as positive.

Category	Model	Acc.	Prec.	Recall	F1	MCC
Contact	LR	52.5%	58.7%	49.1%	53.5%	5.89%
	biLSTM	80.5%	80.5%	96.6%	88.6%	58.3%
	BERT	82.7%	89.1%	85.6%	91.2%	62.5%
Device	LR	76.8%	86.9%	85.9%	86.4%	7.10%
	biLSTM	81.6%	97.4%	82.9%	88.7%	46.7%
	BERT	84.3%	83.4%	97.9%	89.8%	53.4%
Location	LR	58.6%	72.3%	54.8%	62.4%	19.1%
	biLSTM	92.7%	92.7%	90.6%	91.7%	58.6%
	BERT	90.1%	98.3%	85.6%	91.2%	62.0%

Table 5.4: Performance comparison on LR, PPWE-biLSTM, and BERT

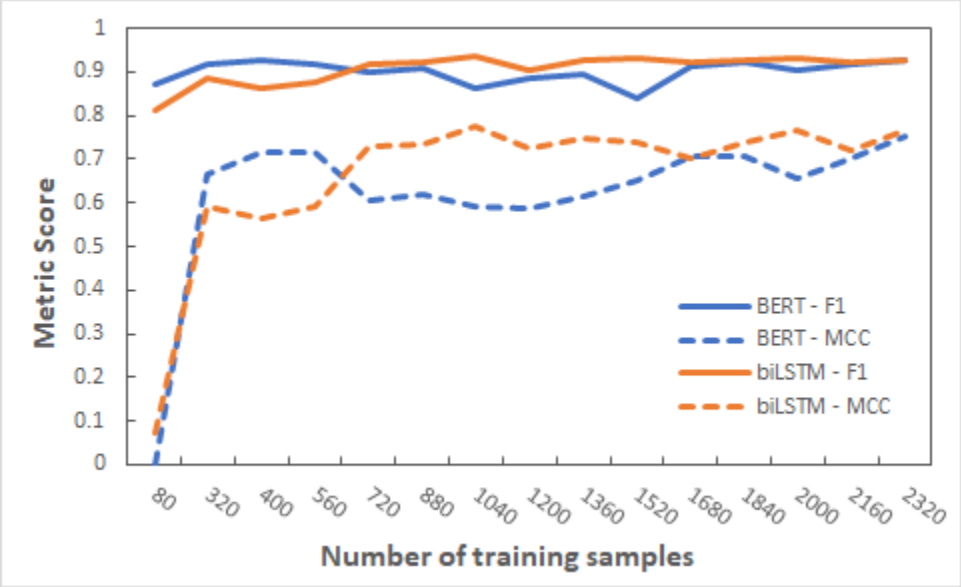

Figure 5.1: F1 and MCC for PPWE-biLSTM and BERT models trained on contact segments

		Acc.	Prec.	Recall	F1	MCC
OPP	LR	94.9%	94.9%	99.9%	97.4%	0.00%
	biLSTM	97.9%	97.9%	100%	98.9%	0.00%
APP	LR	63.6%	75.7%	50.9%	60.9%	31.3%
	biLSTM	90.7%	87.9%	95.8%	90.8%	82.3%
CPPS	LR	58.6%	72.3%	54.8%	62.4%	19.1%
	biLSTM	92.7%	92.7%	90.6%	91.7%	58.6%

Table 5.5: Performance of location classifiers trained on LR and PPWE-biLSTM models

		Acc.	Prec.	Recall	F1	MCC
Opp	LR	88.9%	91.2%	96.5%	93.8%	43.3%
	biLSTM	97.9%	97.9%	100%	98.9%	0.00%
App	LR	64.6%	68.5%	67.3%	67.9%	28.6%
	biLSTM	87.7%	96.9%	82.0%	88.5%	76.9%
CPPS	LR	52.5%	58.7%	49.1%	53.5%	5.89%
	biLSTM	88.6%	83.5%	95.3%	88.7%	75.8%

Table 5.6: Performance of contact classifiers trained on LR and PPWE-biLSTM models

		Acc.	Prec.	Recall	F1	MCC
Opp	LR	96.9%	96.9%	99.9%	98.5%	0.00%
	biLSTM	100%	100%	100%	100%	0.00%
App	LR	59.6%	64.2%	61.8%	62.9%	18.6%
	biLSTM	88.6%	86.2%	90.6%	88.2%	74.6%
CPPS	LR	76.8%	86.9%	85.9%	86.4%	7.10%
	biLSTM	81.6%	97.4%	82.9%	88.7%	46.7%

Table 5.7: Performance of device classifier trained on LR and PPWE-biLSTM models

5.4 Evaluation of Querying Strategies

Moving to the performance evaluation of active learning, the following experimental setups are used in all tests related to this topic. We specify *BATCH_SIZE*=20 for the initial boot-strap phase which

is non-AL, and *BATCH_SIZE*=8 for the stage two active learning phase even though the number of new instances labeled by crowdsourcers may not be a constant value. A large training pool leads to a significant amount of memory consumption during training. For testing efficiency, we limit the unlabeled training pool $\mathbb{X}^U$ to contain $12K$ segments, approximately $4K$ for each category, randomly selected from the 52K policies crawled from Google Play. Note that the actual labeled training set contains fewer segments, as some instances queried by the learner do not pass the AT and fail to be aligned. Unless specified otherwise, we repeat each experiment three times and calculate the average accuracy.

We conduct two sets of experiments using different validation data. Both test sets are generated from the APP-350 corpus, and consist of 300 labels for each category. The difference is that labels in *Test_set_1* are randomly selected while *Test_set_2* is selected to be a balanced set. The former is used to evaluate the model accuracy against the APP-350 dataset, whereas the latter represents a generalized *true* classification accuracy.

For the following experiments, we boot-strap our classifiers with 100 labeled segments. The initial data are prepared by random selection from the unlabeled pool $\mathbb{X}^U$, and are labeled and consolidated using the rule of majority. The labeled training data that are ready to be used are denoted as $\mathbb{X}^L$. In each active learning iteration, the active learner queries new instances from $\mathbb{X}^U$. The selected instances are labeled by Turkers and the aligned labels will be added to $\mathbb{X}^L$. Note that the percentage of negative samples in the initial $\mathbb{X}^L$ differ from one category to another: contact has the highest negative sample rate, 31%, whereas the percentage is 23% for location, and only 8% for device.

In addition to the four querying strategies mentioned in Section 4.3.2, we also include a baseline non-AL model **BASE** for comparison, which uses random sampling to obtain new instances. In both sets of experiments, we measure the performance of querying algorithms using F1 score and MCC, and conduct the same experiments for all three categories. We note that **EER** does suffer from a much slower training speed compared to other methods, as it requires a complete walk-through of all instances to calculate the expected error for each active learning iteration. However, the classifier training time is still overshadowed by the wait time for crowdsourcing tasks, which could take days to complete. However, we point out that if we continued our experiments and labeled more segments, the training time would continue to grow while the labeling time will stay constant.

Figure 5.2 and Figure 5.3 compare the F1 and the MCC performance for different querying strategies in location classifiers evaluated on *Test_set_1*. In both figures, each querying algorithm is associated with individual scores presented by dots, and an estimated trendline based on its average performance scores. We observe that all active learning models outperform the baseline. As a reference, while the active models converge faster, **BASE** reaches an F1 of 98.3% with more than 2200 labels, denoted as the *converged F1 value*. Figure 5.2 shows that, as the most effective active learning model, **BMU** reaches an F1 of 93.4%(95% of the converged F1 value) with 191 labels while **BASE** needs 375 for the same score. That is, the best active learning model uses 50.93% of the training labels used by **BASE** to achieve the same result. Similarly, **BMU** is able to achieve $F1 = 97.3\%$ (99% of the converged F1 value) using 749 labels, which is only 48.97% of the number of training samples used in **BASE**.

To evaluate how much saving in training labels there is when using active learning, we introduce two *Percentile Scores* (PS): *PS_high* and *PS_low*, representing the 90th percentile and 85th percentile for MCC, and the 99th percentile and 95th percentile for F1 score, since F1 values converge faster. We also define Training Effort Percentage (TEP) for each Percentile Score (PS) as $TEP = \frac{n_{AL}}{n_{BASE}}$, where

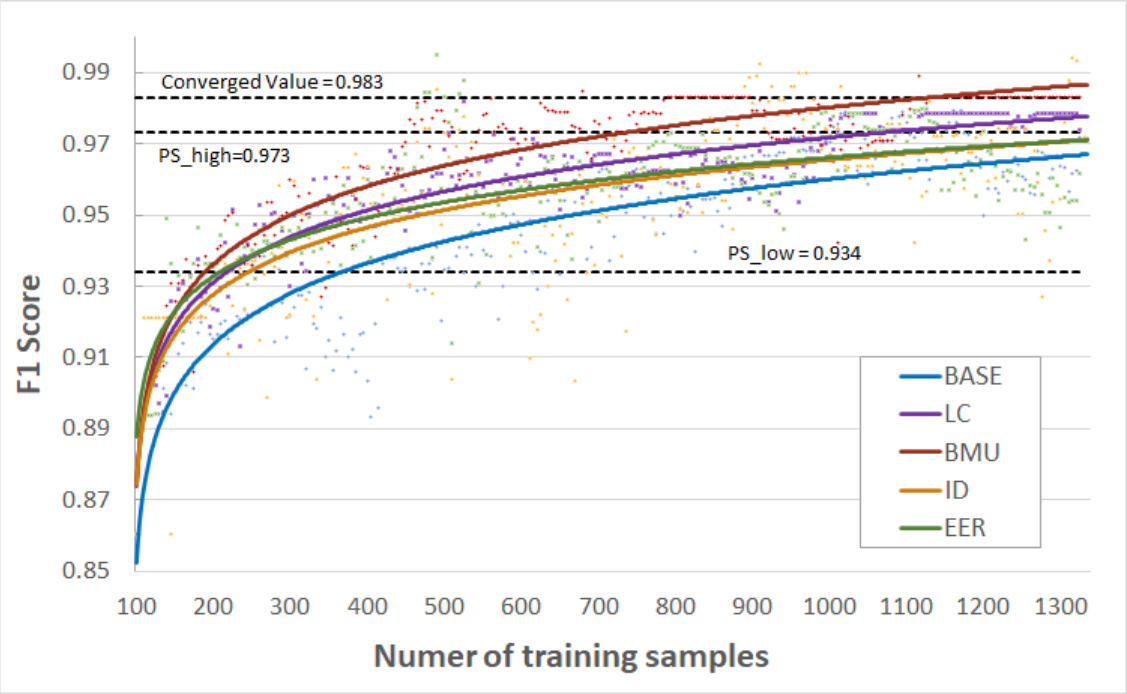

Figure 5.2: F1 score of location classifiers with different querying strategies (Test_set_1)

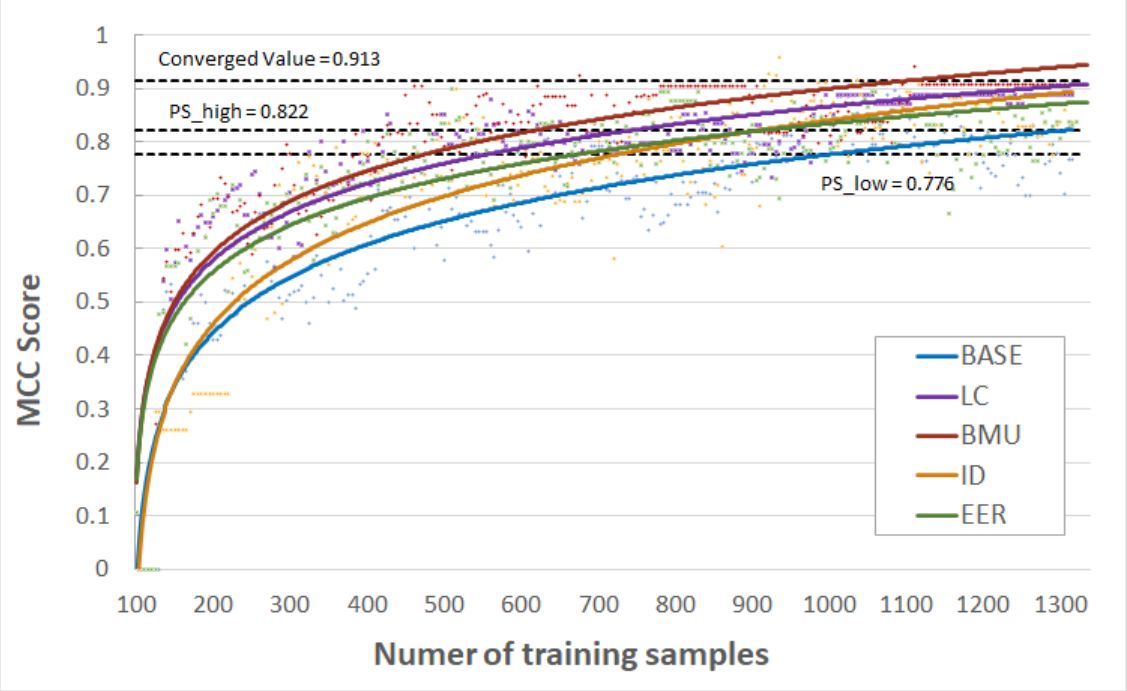

Figure 5.3: MCC score of location classifiers with different querying strategies (Test_set_1)

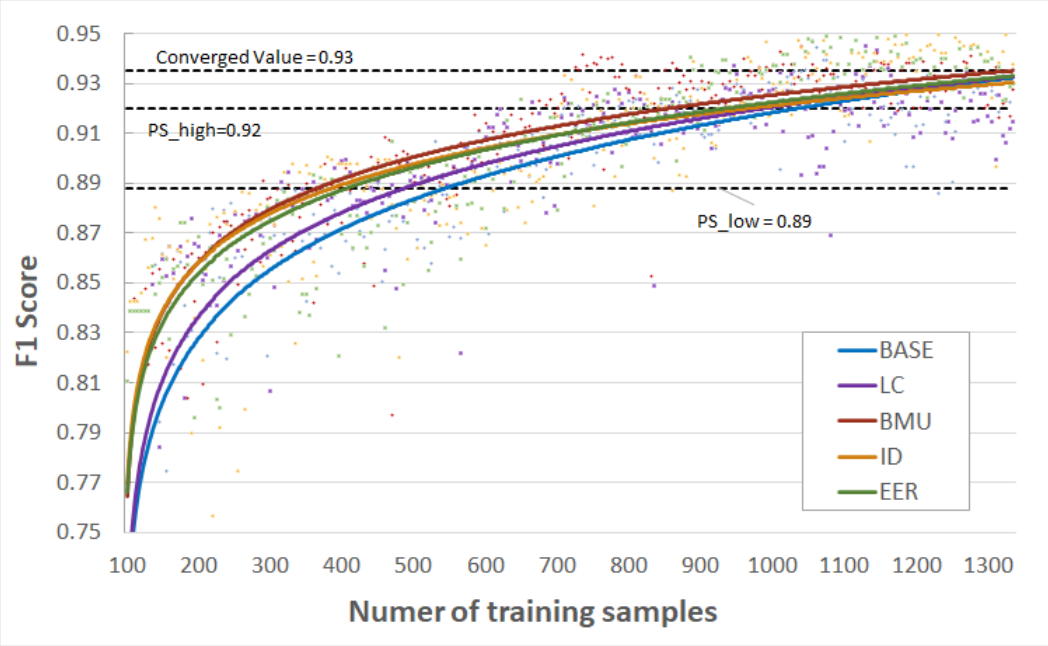

Figure 5.4: F1 score of contact classifiers with different querying strategies (Test_set_1)

n_{BASE} is the number of training samples used in **BASE** to achieve a specific performance goal, and n_{AL} is that number of the *best* case active learning model. A lower score value indicates a more effective active learning algorithm. That is, a querying strategy uses fewer training samples to achieve the same performance.

Going back to the example of Figure 5.2, for location classifiers, the 48.97% we calculated is the TEP of PS_{low} for F1 score, whereas its PS_{low} is 50.93%, which are also shown in Table 5.8. Figure 5.4 and 5.5 show the graphical results for contact classifiers, and 5.6 and5.7 for device. In general, while all active learning querying models outperform **BASE**, **BMU** has the best performance among all. We also observe that, compared to the baseline, the largest amount of training effort saved when using active learning models occurs in the *device* classifiers, where the initial training set is the most imbalanced. The improvements in contact classifiers are relatively small compared to the other two categories, as the contact dataset contains more negative samples. Further evaluations on active learning models and class imbalance will be presented in Section 5.7.

Category	Metric	Converged Value	PS_low	PS_high
Contact	MCC	0.78	70.87%	71.43%
	F1	0.93	85.37%	85.63%
Device	MCC	0.90	27.11%	26.53%
	F1	0.99	35.00%	38.89%
Location	MCC	0.91	42.85%	31.88%
	F1	0.98	48.97%	50.93%

Table 5.8: Performance improvement: AL vs. non-AL (Test_set_1)

We also make an observation from Table 5.8 that the most effective active learning models are the ones trained on the device category, whereas the contact active learning classifiers do not save as

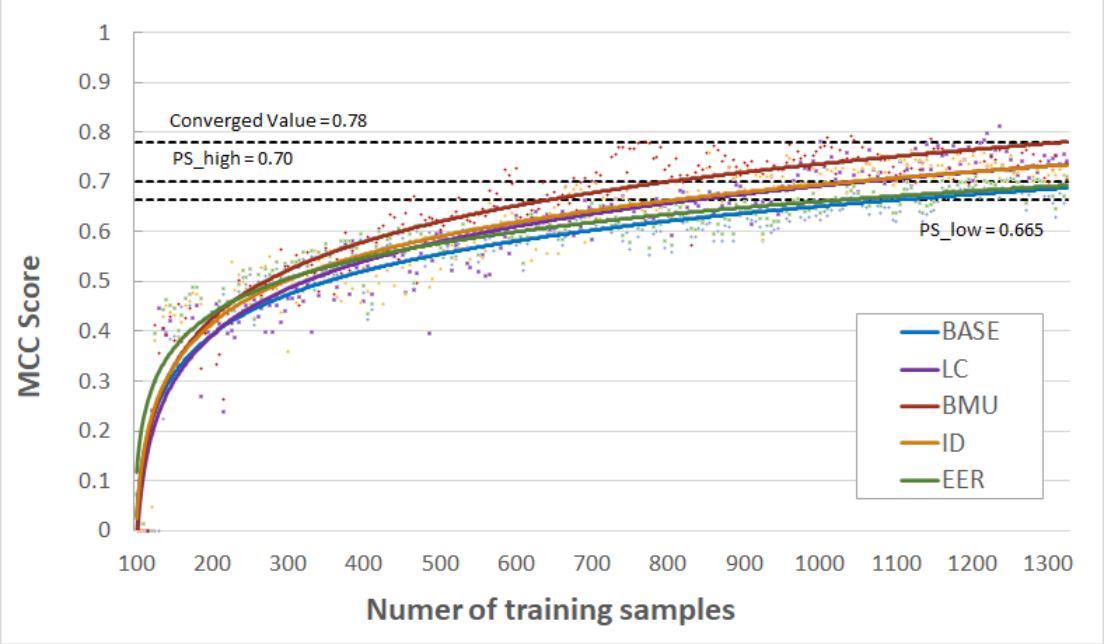

Figure 5.5: MCC score of contact classifiers with different querying strategies (Test_set_1)

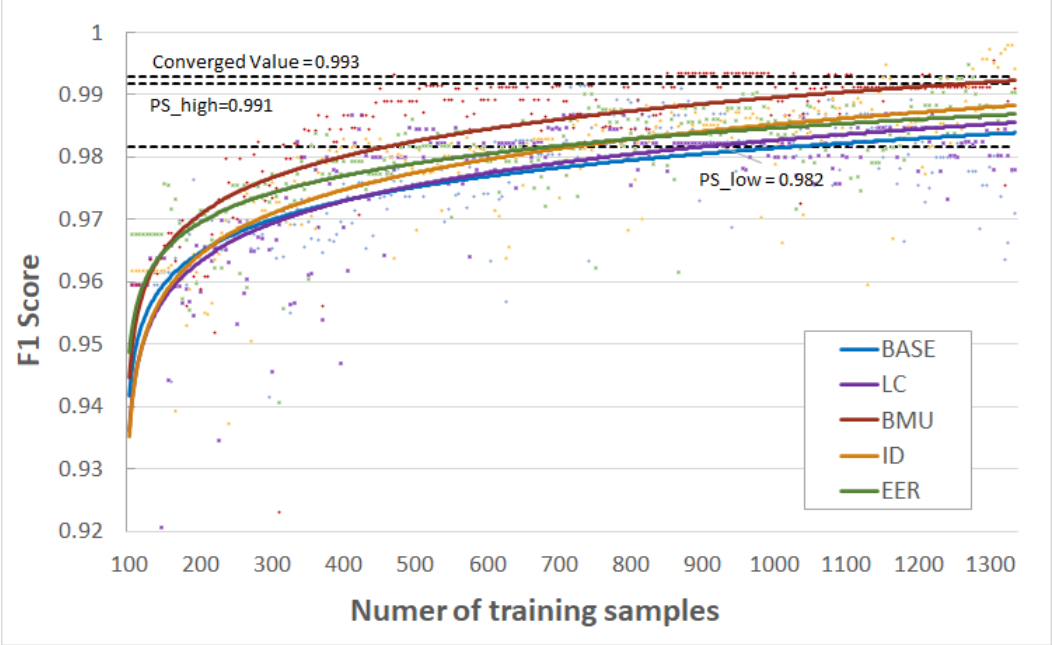

Figure 5.6: F1 score of device classifiers with different querying strategies (Test_set_1)

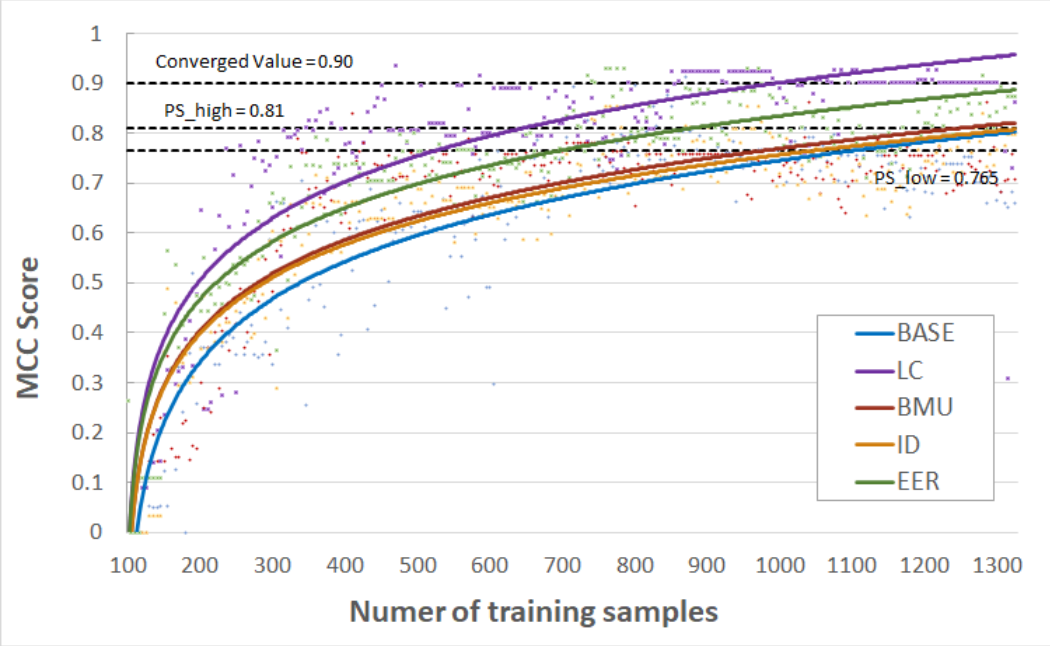

Figure 5.7: MCC score of device classifiers with different querying strategies (Test_set_1)

much labeling effort. This is reasonable since privacy policy segments on device information tend to be simpler and more straightforward than contact and location. The average length of device segments is also slightly shorter than the other two. As an example, indications for device labels are mainly "device information" and "IP address", whereas the contact category contains a lot more relevant keywords, such as email, phone number, contact book, different social network accounts such as Facebook, Twitter, or Google. Moreover, the keyword "contact" may also refer to other meanings. For instance, most privacy policies include a section for users to contact the App developers. Such sentences are also associated with email address or phone number, making it difficult for the classifiers to differentiate such segments from the actual data practice of collection/usage of contact information. As a result, it requires more labeling effort to achieve a nearly converged accuracy.

In the second set of experiments, we evaluate the models using the balanced validation set *Test_set_2*. As the initial training set has significant class imbalance issues, performance scores on *Test_set_2* converge slower than that of *Test_set_1*. We therefore focus on the early stage of training and investigate the amount of labeling effort saved by active learning models to achieve an early performance goal. We set the stopping criteria for all classification models to be: $MCC > 0.2$ and $F1 > 70\%$. We run experiments on the three categories, and observe similar results as those for *Test_set_1*: All active learning models outperform **BASE**, with **BMU** being the faster model to reach the stopping goal. Figure 5.8 summarizes the difference in labeling effort when using active learning and non-AL **BASE** models. We conclude that our active learning model is able to achieve the same performance with fewer training labels, whether the test sets are perfectly balanced or reflect the actual data distribution of the input space.

Similar to the previous experiments, the TE percentage is calculated using the ratio of n_{AL} and n_{BASE}. However, we observe a difference in these TE values: the device active learning classifier has a higher TE than the other two categories, while the TE for contact classifier drops to the lowest. This is

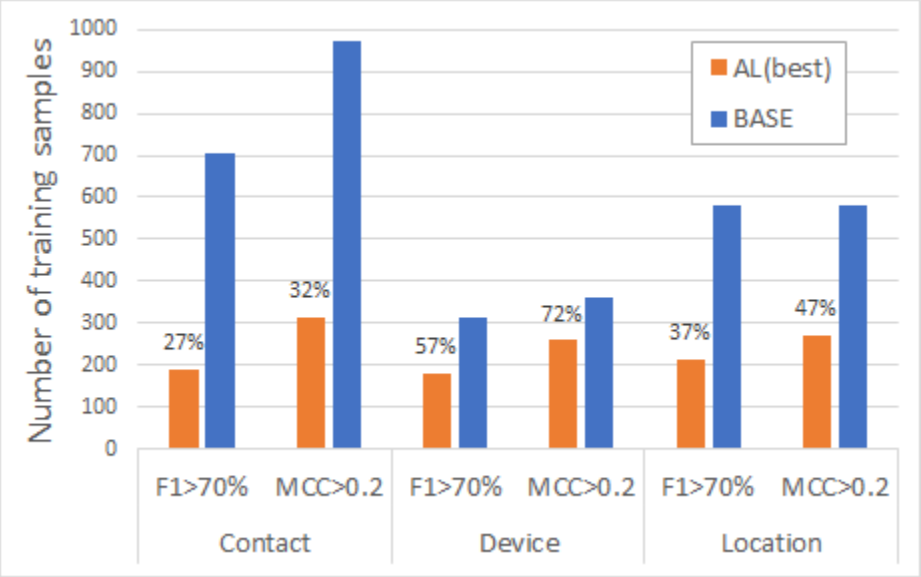

Figure 5.8: Training effort on Test_set_2: AL vs. non-AL

also reasonable, as the second experiment is done in the early stage of a training process. Looking at the actual number of training samples shown in Figure 5.8, we observe that classifiers for all data categories require approximately 200 labels to be able to surpass a 70% F1 score, and 280 labels to achieve an MCC higher than 0.2. On average, to achieve the same performance, Calpric only needs 45.3% of the training effort used by the original non-AL models.

5.5 Impact of Acceptance Threshold

To investigate the relationship between AT values and the learning speed, we experiment on three different AT values: 100%, 80%, and 60% using the same active learning model. The testing model is implemented with **BMU**, which was shown to be the best querying strategy in the previous section. We aim to introduce 30 new labels in each active learning iteration, therefore we set the learner to query 42 unlabeled segments per iteration, according to the average alignment rate of 73%. After the annotation, Calpric downloads the resulting labels and process them with the rule of majority. It consolidates these labels only when the number of majority votes reaches the testing threshold. For instance, if the current testing $AT = 1.0$, we train our classifiers using only the perfectly aligned labels (all workers agree on the same label), and discard the remaining ones where AR $<$ AT.

We define each crowdsourcing label obtained from Amazon mTurk as one unit of labeling effort (LE). In other words, LE takes into account the segments sent for labeling but do not reach the minimum AT. In this experiment, we allocate 8000 units of LE to each AT value during the active querying phase. That is, the active learning model is able to query up to 1600 different unlabeled segments, each labeled by five Turkers, even if some of them will not align. We set the boot-strap LE at 100 for each classifier, which means they will probably end up with different numbers of initial training labels because of the different AT values. We train classifiers separately using the post-processed aligned labels according to each pre-set AT, and evaluate their performance using the balanced validation set: *Test_set_2*.

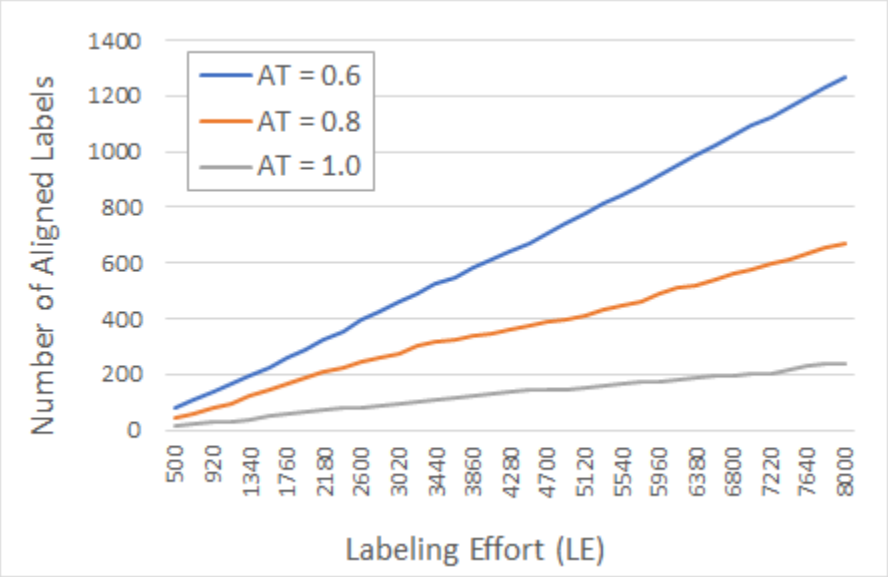

Figure 5.9: Aligned labels vs. LE for different AT on contact classifiers

As an example, Figure 5.10 shows the trend of learning speed vs. F1 Score performance for contact classifiers. In general, the scores in all three cases are relatively closed to each other. However, $AT = 1.0$ is less stable compared to others, as shown by the large fluctuations, because its training set is significantly smaller than that of $AT = 0.8$ and $AT = 0.6$. Figure 5.11 shows the trend of learning speed vs. MCC performance for the same contact classifiers. Both $AT = 0.8$ and $AT = 1.0$ outperform $AT = 0.6$. While they share a similar performance trend, $AT = 1.0$ shows the instability similar to its F1 performance. We include a figure showing the relationship between the number of aligned labels and the amount of LE allocated in Figure 5.9, indicating how many labeled instances are generated from the same LE when the AT values vary. We observe a pattern in which the lower the AT is, the more aligned labels we receive from the same amount of allocated LE. On the other hand, despite having a large training set, $AT = 0.6$ results in the worst performance. This is because the low AT value allows labels with low confidence to be added to the dataset, resulting in unwanted label noise. We observe that, in general, the use of active learning results in a larger improvement in MCC performance compared to the F1 score, because active learning in our study naturally targeted the class imbalance problem, and MCC is a direct measurement of how the classifiers are making *balanced* predictions. In practice, to improve the MCC performance, one should train the model using a more balanced dataset. On the other hand, the most effective way to increase the F1 score is to train models using a larger training set. Our AT experiments are done on relatively small training sets with an average size of 847 aligned labels. As a result, the improvement in F1 will be rather insignificant.

We also conduct experiments on the two other categories and the results are similar. Figure 5.12 and 5.13 show the trends of device classifiers. Note that the F1 score of $AT = 0.6$ fluctuates at the initial stage. We suspect that a few mislabeled segments are introduced to the training process, confusing the classifiers and leading to undesired results. Similarly, Figure 5.14 and 5.15 show the performance trends of location classifiers. We observe a slight difference in the F1 result: both $AT = 0.6$ and $AT = 1.0$ show stable trends whereas $AT = 0.8$ fluctuates in a larger range. However, the overall trend is increasing

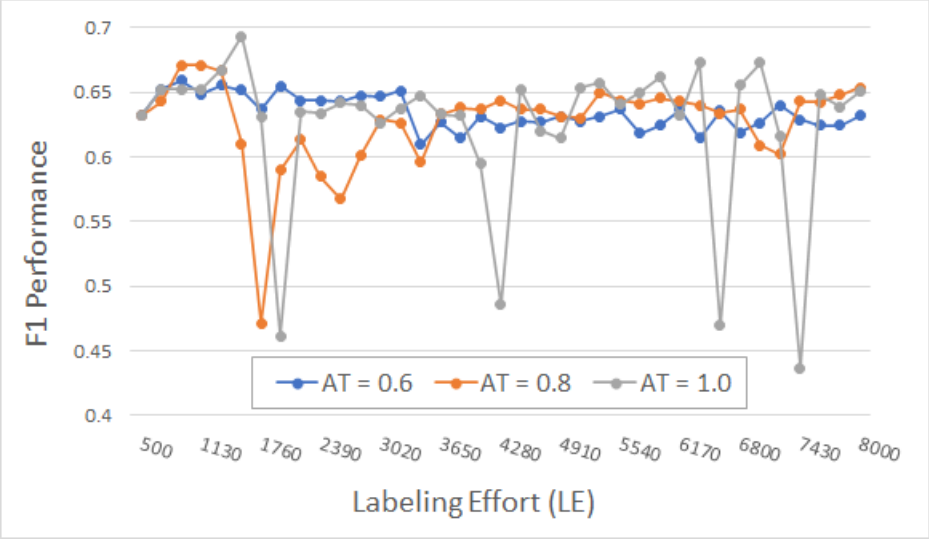

Figure 5.10: F1 performance for different AT on contact classifiers

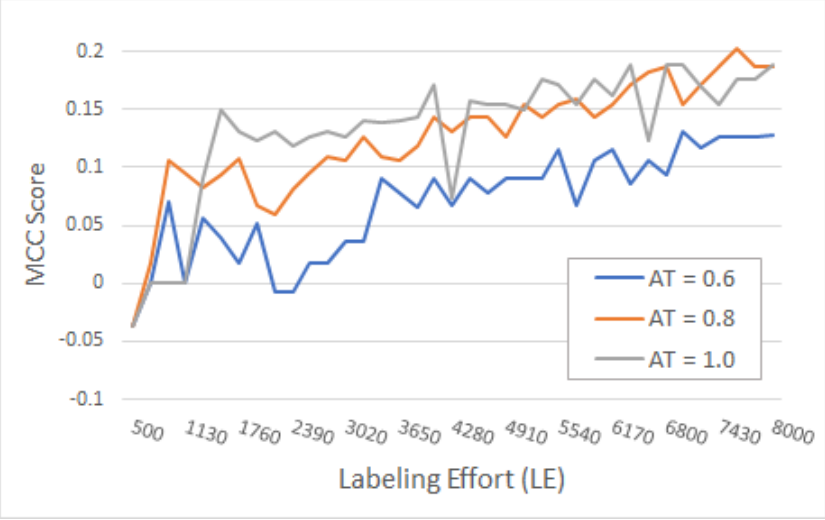

Figure 5.11: MCC performance for different AT on contact classifiers

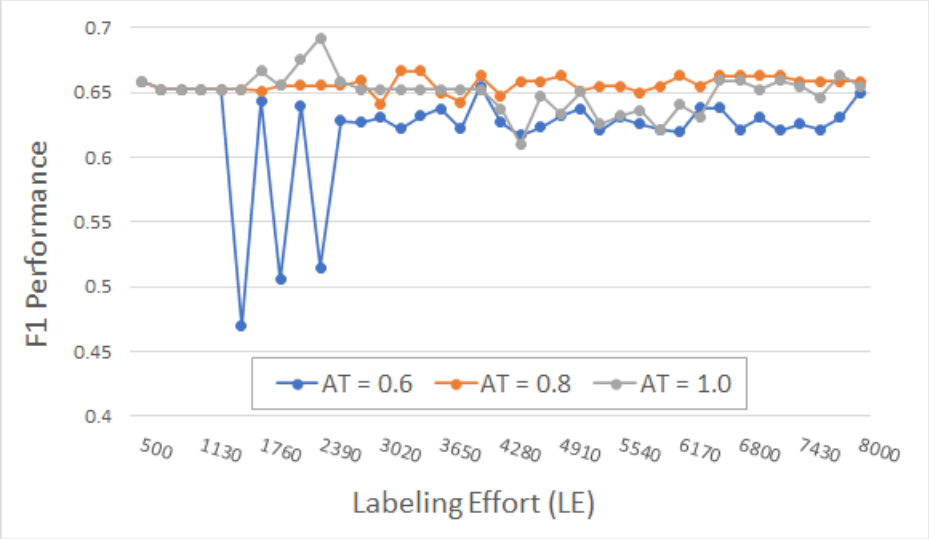

Figure 5.12: F1 performance for different AT on device classifiers

Category	AT	Aligned Labels	F1	MCC
Contact	0.6	1266	0.6318	0.1273
	0.8	673	0.6535	0.1870
	1.0	239	0.6503	0.1889
Device	0.6	1308	0.6499	0.0909
	0.8	885	0.6583	0.1059
	1.0	547	0.6547	0.1147
Location	0.6	1333	0.6795	0.1167
	0.8	829	0.6999	0.1596
	1.0	542	0.6455	0.0679

Table 5.9: AT performance comparison using 1600 labels

and so does its MCC performance. In addition, the experiments are performed using a relatively small amount of data, some minor fluctuations during the training process should be negligible.

The full performance comparison is summarized in Table 5.9. Although in most cases $AT = 1.0$ and $AT = 0.8$ outperform $AT = 0.6$, we do observe a difference in the results for the location classifiers only: When AT is set to 1.0, the both F1 and MCC performance drop to the worst among all three AT values.

While other classifiers are trained on samples with similar NSR despite being set to different AT, the location classifiers are not. We calculated the standard deviation for NSR in the three categories: 0.0425 for contact, 0.0301 for device, and 0.1239 for location. Specifically, labels consolidated using $AT = 0.6$ and $AT = 0.8$ contain an average of 52.0% and 52.1% negative samples, respectively, whereas $AT = 1.0$ only has an NSR of 25.5%. Furthermore, as the AT value increases, the total number of aligned labels decreases, together resulting in only 138 negative labels for the 1.0 classifier. It is possible that the some crowdsourcers did not agree on labels of potential negative samples. Because we set AT to be 100%, any misalignment will result in the failure to produce to pass the AT. These unaligned labels are not allowed to be added into the training set. Since we test on a relatively small $\mathbb{X}^U$ (4K unlabeled segments), we may have run out of negative samples quickly, and the active learner would not

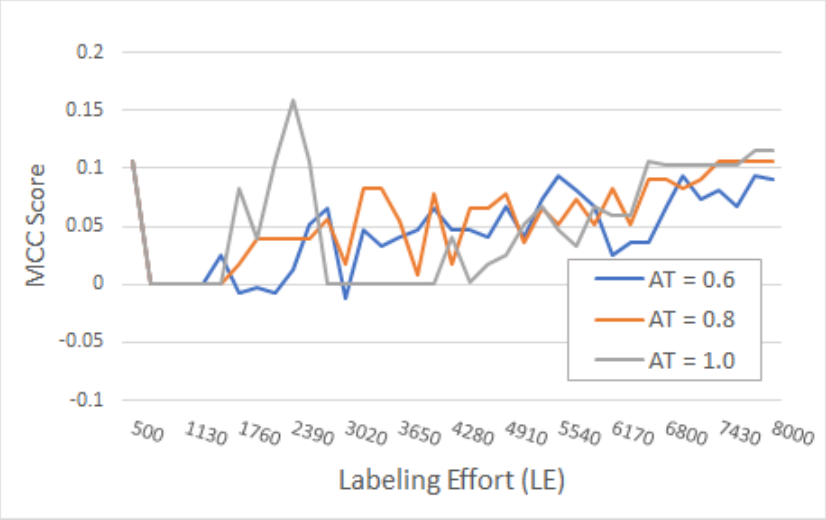

Figure 5.13: MCC performance for different AT on device classifiers

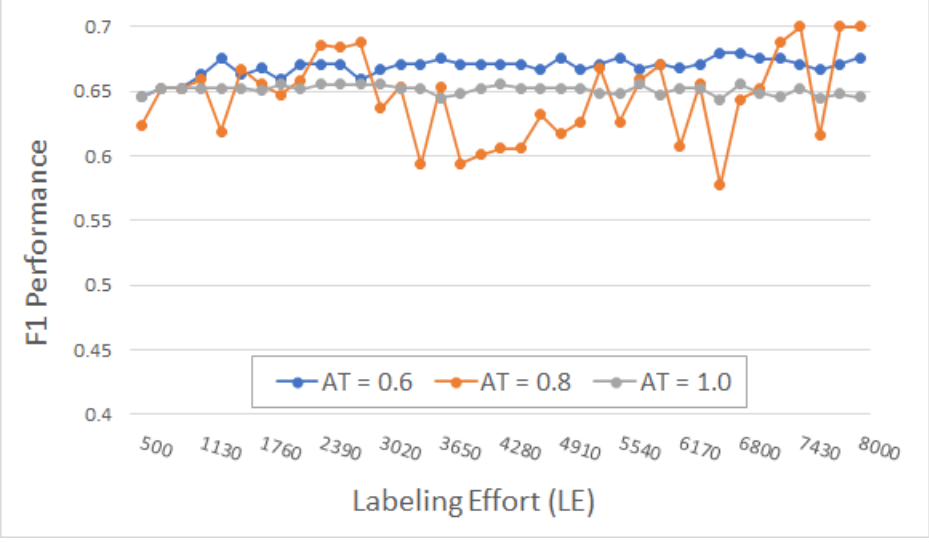

Figure 5.14: F1 performance for different AT on location classifiers

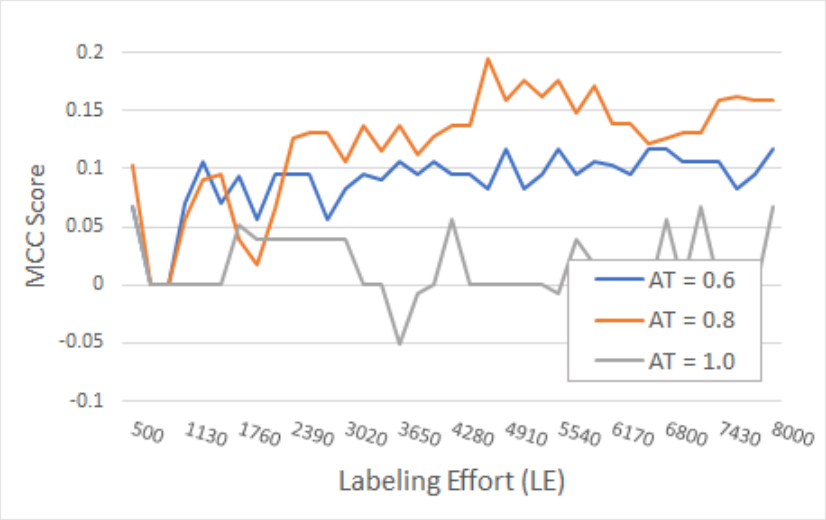

Figure 5.15: MCC performance for different AT on location classifiers

able to query more informative segments to be labeled. Being trained on an imbalanced dataset, the $Location - AT = 1.0$ model suffers from under-fitting, and its F1 accuracy and MCC reflect this. We explore re-labeling methods to handle discarded labels in Section 5.6, and further investigate the impact of negative samples in Section 5.7.

Also note that the AR for $AT = 0.8$ obtained in this experiment seems to be lower than that in previous experiments (73%) in which instances are randomly selected. We suspect this is due to the active selection bias, where the queried labels contain more uncertainty than others.

From the above experiments, we conclude that $AT = 0.8$ is the appropriate pre-set threshold for active learning policy classification systems of a size similar to that of our prototype tool. That being said, given a very large unlabeled training pool and an unlimited amount of LE, $AT = 1.0$ does achieve the performance ceiling, as it has the lowest possibility of introducing additional label noise to the training set.

5.6 Re-labeling strategies

In theory, we should not discard unaligned labels in active learning classification, as they do contain a significant amount of information, which could potentially improve the model performance. As described in Section 4.3.2, we implement two types of methods to handle unaligned labels: Label and Discard (L&D) and Incremental Re-Labeling (IRL). In this section, we compare these strategies using the following setup: active learning models with **BMU** querying strategy and AT value set to 80%. We use the $12K$ training pool and evaluate the classifiers using *Test_set_2*.

We run the IRL model on classifiers based on the three different categories, and allow each classifier to re-label up to 100 segments. The number is denoted as the allocated re-labeling resource. As mentioned, we set the maximum number of labeling iterations to be $N = 3$. Under such circumstances, if we re-label one segment ($N = 2$) and a total of 10 crowdsourcers still could not agree on a single label, we can once again publish this segment to be labeled ($N = 3$), resulting in a total of 15 labels. That being said, if

we publish the same segment twice, we count this as two units used in the re-labeling resource.

We introduce a measurement of how effective the re-labeling process is. The re-labeling Success Rate (RSR) is defined as a ratio of *the number of successfully aligned labels after re-labeling* and *the total allocated re-labeling resource*. We calculate RSR for each classifier. Unfortunately, the results for relabeling are not promising. Out of 100 relabeled segments, only 2 passed the AT after $N = 2$ and only 1 passed the AT after $N = 3$, resulting in an overall RSR of 3%.

One possible reason for such a low RSR is that, because our question surveys are carefully designed and reviewed, with the additional qualification tests, crowdsourcers are able to provide correct labels as long as they pay attention to the questions. Furthermore, if the segment is longer and more complex than usual, the possibility of misinterpretation increases, affecting the successful rate of label alignment. This is proven by our experiment, as most *re-labeled and aligned* labels have longer lengths than the average.

Another consideration is that, some labels are ambiguous by nature. For instance, examples of the *relabeled and failed* include: "your GPS geo-location is not accessed without your consent" and "you will be asked for your permission each time a location-based service is requested to provide you with local search results." It is obvious that the software is trying to access user information in order to perform a certain service. However, users also have the choice to disable such functionalities, or opt out of specific features. Such segments, even when crowdsourcers agree on the label itself, may not reflect the actual legal content. What is worse, these labels may introduce additional data noise and contaminate the training set.

We conclude that developing a more complex re-labeling design may not be an ideal choice to avoid under-fitting. Many of the segments that do not reach AT the first time are likely ambiguous and are unlikely to pass AT if more labels are obtained from Turkers, and thus it is generally a more efficient use of resources to discard such segments. Instead, we should construct a larger unlabeled training pool $\mathbb{X}^U$ for the active learning system to query as many informative labels as possible.

5.7 Percentage of Negative Samples

In previous experiments, we observe that the active learning models not only improve the training speed, but also solve the existing problem of class imbalance in privacy policy classification.

Figure 5.16 shows the trends of NSR for the labeled training set $\mathbb{X}^L$ using different querying strategies. We observe steep rises in **LC**, **BMU**, and **EER**, and a slightly higher NSR in **ID** than that of **BASE**. This is reasonable, as the initial boot-strap training set contains fewer negative samples. The classifiers are more uncertain on such instances, and query more of them during each active learning iteration. We note that all NSR converge back to the initial value as the size of $\mathbb{X}^L$ increases. This is due to our limited $\mathbb{X}^U$, which contains only $4K$ unlabeled segments that can be potentially queried. As mentioned, the number of negative samples in privacy policies tend to be much fewer than that of the positive ones. Thus, it is likely that all negative samples are used up as the training process continues.

Figure 5.17 confirms this hypothesis. All active learning models query more negative instances than the random baseline model. As a result, while there are still remaining negative samples in $\mathbb{X}^U$ for **BASE**, other models run out of negative samples in the first few hundreds of learning iterations. For future work, it will be crucial to obtain a reasonably large $\mathbb{X}^U$ before training the active learning models

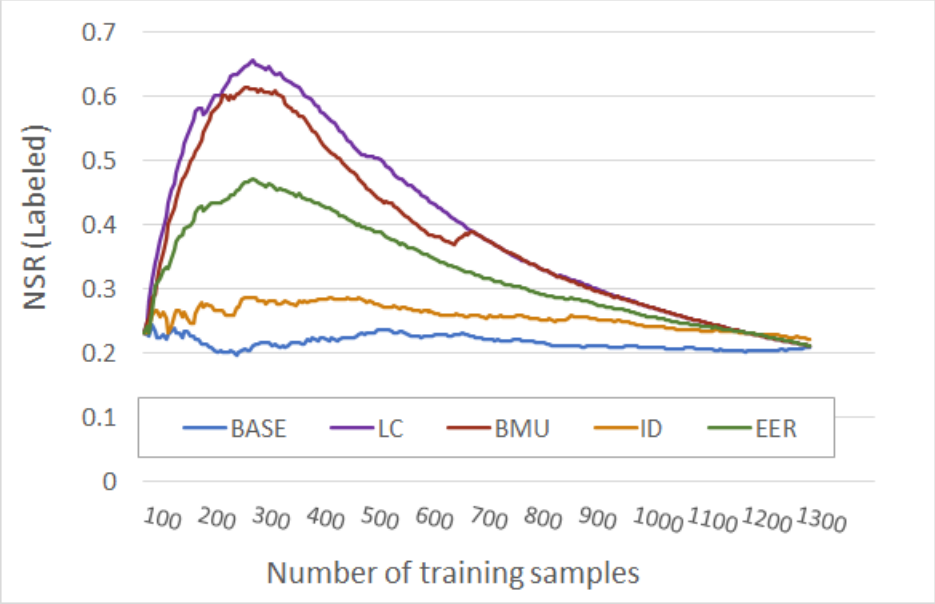

Figure 5.16: Location classifiers: train set NSR for different querying strategies

to ensure the set contains an adequate number of negative samples. We can approximate the amount of negative training labels needed to achieve certain goals, and back calculate the required size of unlabeled training pool $\mathbb{X}^U$ using the estimate percentage of NSR in the pool.

Similarily, for device and contact categories, we perform the same evaluation. Figure 5.18 and 5.19 show the comparison of NSR for device classifiers, whereas Figure 5.20 and 5.21 highlight the trends of contact classifiers.

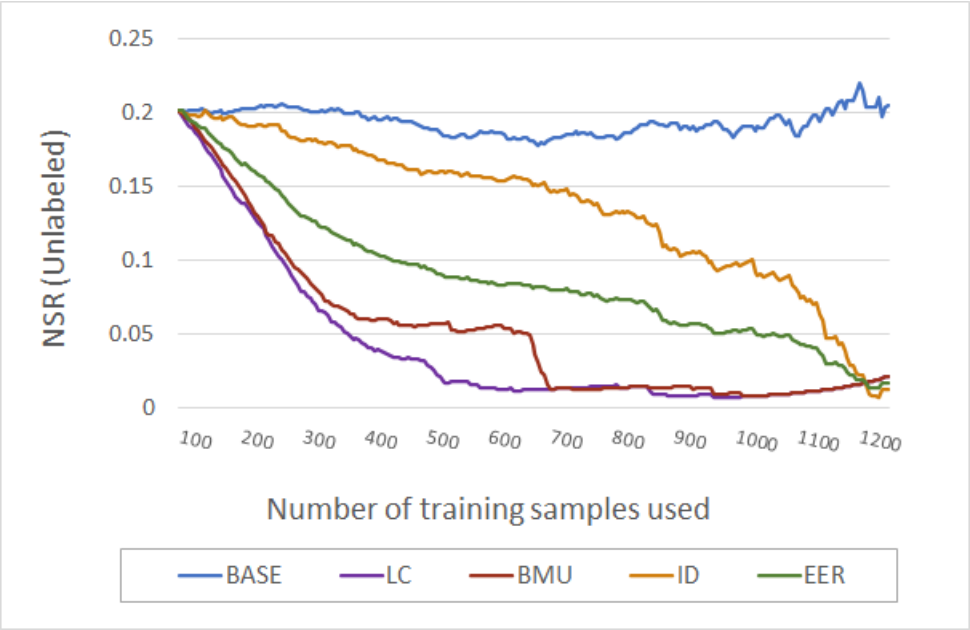

Figure 5.17: Location classifiers: unlabeled pool NSR for different querying strategies

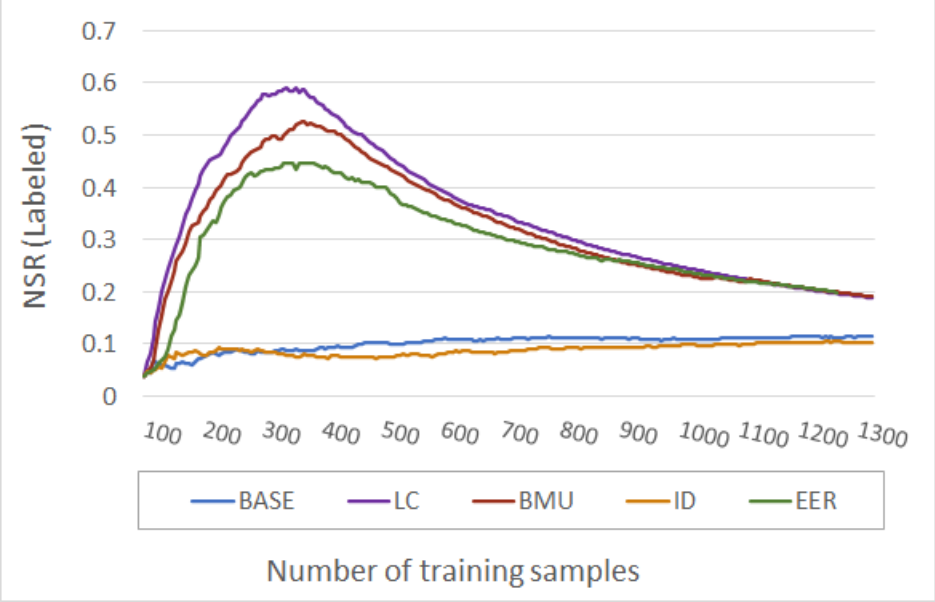

Figure 5.18: Device classifiers: train set NSR for different querying strategies

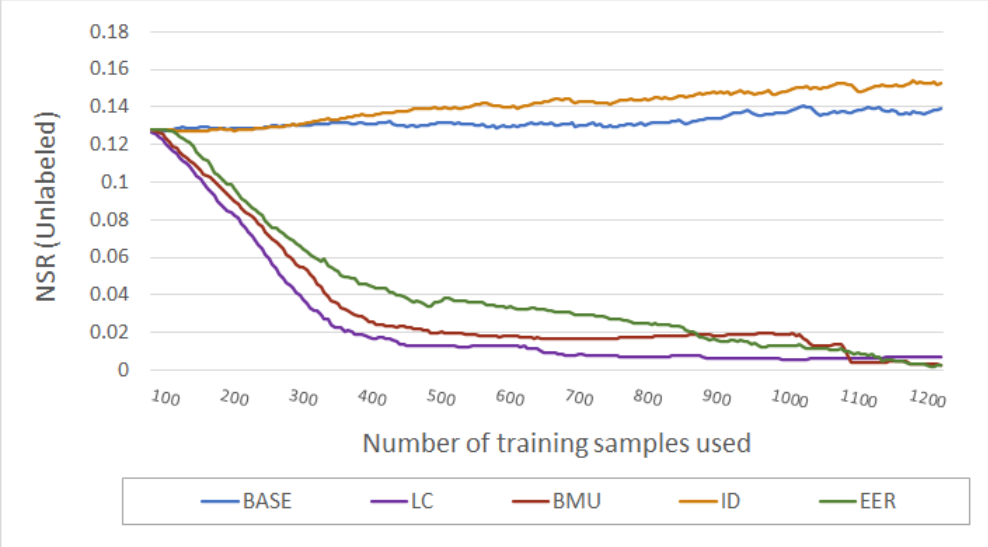

Figure 5.19: Device classifiers: unlabeled pool NSR for different querying strategies

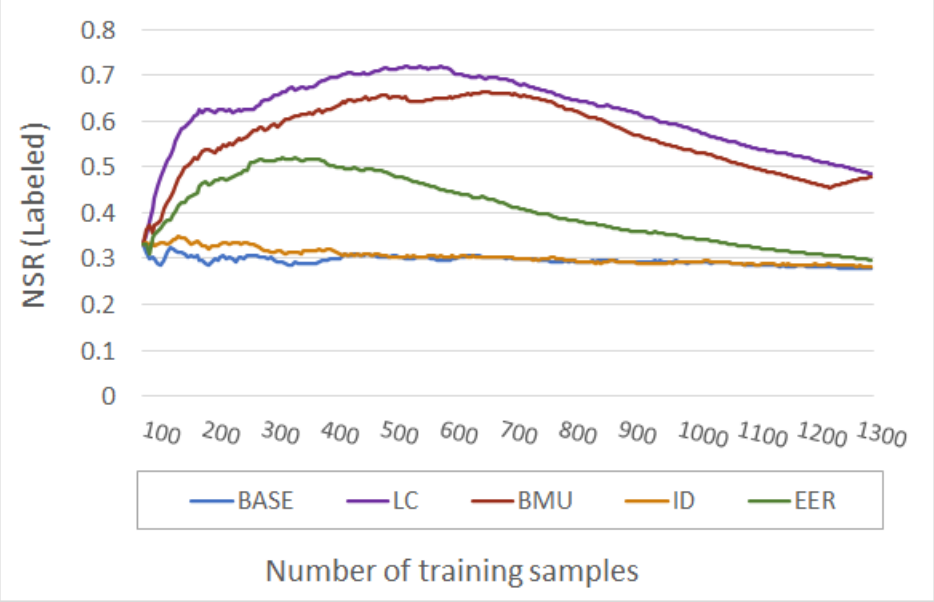

Figure 5.20: Contact classifiers: train set NSR for different querying strategies

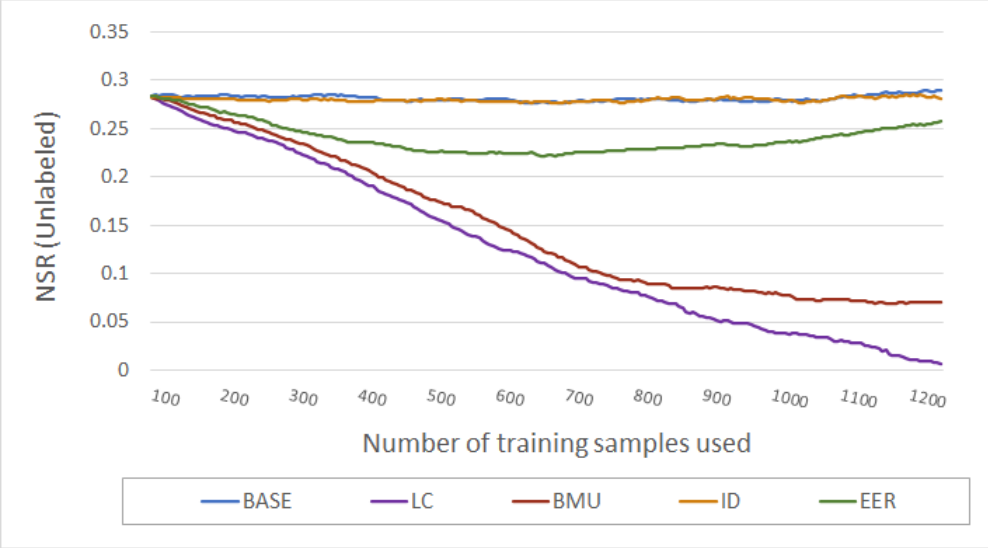

Figure 5.21: Contact classifiers: unlabeled pool NSR for different querying strategies

Chapter 6

Limitations, Future Works and Conclusion

6.1 Limitations and Future Works

As the first automated privacy policy analyzer that leverages active learning techniques, we designed and built Calpric as a prototype tool. As stated previously, in this thesis, we focus solely on data practices related to first party collection/use, and our classifiers are trained and evaluated on privacy policy segments of three selective data categories: contact, location, and device information. As a result, instead of being able to handle all privacy policy texts captured from the existing application market, Calpric is only capable of analyzing a limited set of data collection. Fortunately, previous studies show that it is not difficult to integrate machine learning based privacy policy tools from one data category to another, as the theory behind the scene is similar for all categories [53]. We look forward to expanding our binary classifiers to more categories, as well as to a second-level in which data practices are classified into First Party Collection/Use and Third Party Sharing. We aim to develop an automated report tool that can take any raw privacy policy texts and output a human-readable report to summarize all user data mentioned in the policies.

As another future direction, to provide guidelines for technology business to develop well-structured privacy policies, and support legal regulators to identify problematic statements in privacy policies, we aim to identify the ambiguous phrases in policy texts and to avoid them in future cases. We want to design an automated classification tool similar to Calpric but is capable of searching for vague sentences throughout a privacy policies and provide suggestions for modifications that make the statement clearer and more straightforward. This additional feature is especially useful for topics that address third party sharing, which contain a lot more ambiguous phrases than the first party data practices, as we discovered.

6.2 Conclusion

Privacy policies are mandatory documents of online application and websites required by privacy legislation. Despite they being the major channel for business to communication with users on collection,

use and sharing of their private information, few users are willing to spend time and read through the entire documents. Automated privacy policy analysis tools were proposed to solve such problems. However, machine learning based classifiers require a large amount of training data to be labeled in order to achieve a high accuracy. It is very expensive and time-consuming to label privacy policies by legal experts. In this thesis, we propose Calpric, the first crowdsourcing active learning framework that performs automated classification of privacy policies with high accuracy, but using fewer than 45.3% of the training labels needed for non-active classifiers. With active learning, our tool proactively selects the most informative batches of policy segments from the unlabeled training pool, increasing the benefit of each label introduced to the training set.

Calpric's use of active learning also addresses naturally occurring class imbalance in unlabeled privacy policy datasets as there are many more statements stating the collection of private information than stating the absence of collection. By selecting samples from the minority class for labeling, Calpric automatically creates a more balanced training set.

To conclude, Calpric opens the opportunity for privacy policy classification tools to achieve high accuracy with a limited labeling budget. It helps application users and regulators to analyze data practices in policies without reading through copious amounts of text.

Bibliography

[1] "Amazon Mechanical Turk API Reference - Amazon Mechanical Turk." [Online]. Available: https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/Welcome.html

[2] "HTML Standard." [Online]. Available: https://html.spec.whatwg.org/multipage/introduction.html

[3] "LEGO IDEAS - Product Idea Guidelines." [Online]. Available: https://ideas.lego.com/guidelines

[4] "Number of apps in leading app stores." [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[5] "User Data - Play Console Help." [Online]. Available: https://support.google.com/googleplay/android-developer/answer/9888076

[6] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python*, 1st ed. Beijing ; Cambridge [Mass.]: O'Reilly, 2009, oCLC: ocn301885973.

[7] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, Dec. 2017. [Online]. Available: https://www.mitpressjournals.org/doi/abs/10.1162/tacl_a_00051

[8] T. N. Cardoso, R. M. Silva, S. Canuto, M. M. Moro, and M. A. Gonçalves, "Ranked batch-mode active learning," *Information Sciences*, vol. 379, pp. 313–337, Feb. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0020025516313949

[9] ——, "Ranked batch-mode active learning," *Information Sciences*, vol. 379, pp. 313–337, Feb. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0020025516313949

[10] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, p. 6, Dec. 2020. [Online]. Available: https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-019-6413-7

[11] E. Costante, Y. Sun, M. Petković, and J. den Hartog, "A machine learning solution to assess privacy policy completeness: (short paper)," in *Proceedings of the 2012 ACM workshop on Privacy in the electronic society - WPES '12*. Raleigh, North Carolina, USA: ACM Press, 2012, p. 91. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2381966.2381979

[12] A. Culotta and A. Mccallum, "Reducing labeling effort for structured prediction tasks." in *Proceedings of the National Conference on Artificial Intelligence*, vol. 2, 01 2005, pp. 746–751.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[14] G. Dias, E. Alves, and J. G. P. Lopes, "Topic segmentation algorithms for text summarization and passage retrieval: An exhaustive evaluation," in *AAAI*, 2007.

[15] Y. Gal, R. Islam, and Z. Ghahramani, "Deep Bayesian active learning with image data," *arXiv:1703.02910 [cs, stat]*, Mar. 2017, arXiv: 1703.02910. [Online]. Available: http://arxiv.org/abs/1703.02910

[16] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM networks," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4. Montreal, Que., Canada: IEEE, 2005, pp. 2047–2052. [Online]. Available: http://ieeexplore.ieee.org/document/1556215/

[17] H. Harkous, K. Fawaz, R. Lebret, F. Schaub, K. G. Shin, and K. Aberer, "Polisis: Automated analysis and presentation of privacy policies using deep learning," *arXiv:1802.02561 [cs]*, Jun. 2018, arXiv: 1802.02561. [Online]. Available: http://arxiv.org/abs/1802.02561

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 770–778. [Online]. Available: http://ieeexplore.ieee.org/document/7780459/

[19] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982. [Online]. Available: http://www.pnas.org/cgi/doi/10.1073/pnas.79.8.2554

[20] Howe and Jeff, "The rise of crowdsourcing," *Wired*, vol. 14, 01 2006.

[21] P. G. Ipeirotis, F. Provost, and J. Wang, "Quality management on Amazon Mechanical Turk," in *Proceedings of the ACM SIGKDD Workshop on Human Computation - HCOMP '10*. Washington DC: ACM Press, 2010, p. 64. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1837885.1837906

[22] N. Joselson and R. Hallen, "Emotion classification with natural language processing (comparing bert and bi-directional lstm models for use with twitter conversations)," Ph.D. dissertation, University of Cape Town, 2019. [Online]. Available: http://rgdoi.net/10.13140/RG.2.2.16482.27844

[23] J. Kim, M. El-Khamy, and J. Lee, "Residual LSTM: Design of a deep recurrent architecture for distant speech recognition," in *Interspeech 2017*. ISCA, Aug. 2017, pp. 1591–1595. [Online]. Available: http://www.isca-speech.org/archive/Interspeech_2017/abstracts/0477.html

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980 [cs]*, Jan. 2017, arXiv: 1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980

[25] C. Kohlschutter, P. Fankhauser, and W. Nejdl, "Boilerplate detection using shallow text features," in *Proceedings of the third ACM international conference on Web search and data mining - WSDM '10*. New York, New York, USA: ACM Press, 2010, p. 441. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1718487.1718542

[26] B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, Nov. 2016. [Online]. Available: http://link.springer.com/10.1007/s13748-016-0094-0

[27] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," *Proceedings on International Conference on Machine Learning*, p. 10, 2015.

[28] D. D. Lewis and J. Catlett, "Heterogeneous uncertainty sampling for supervised learning," in *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 148–156. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B978155860335650026X

[29] F. Liu, S. Wilson, P. Story, S. Zimmeck, and N. Sadeh, "Towards automatic classification of privacy policy text," p. 5, 2018.

[30] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," *arXiv:1605.05101 [cs]*, May 2016, arXiv: 1605.05101. [Online]. Available: http://arxiv.org/abs/1605.05101

[31] M. Lovett, S. Bajaba, M. Lovett, and M. J. Simmering, "Data quality from crowdsourced surveys: A mixed method inquiry into perceptions of Amazon's Mechanical Turk Masters," *Applied Psychology*, vol. 67, no. 2, pp. 339–366, Apr. 2018. [Online]. Available: http://doi.wiley.com/10.1111/apps.12124

[32] A. L. Maas, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning (ICML)*, 2013.

[33] W. Mason and S. Suri, "Conducting behavioral research on Amazon's Mechanical Turk," *Behavior Research Methods*, vol. 44, no. 1, pp. 1–23, Mar. 2012. [Online]. Available: http://link.springer.com/10.3758/s13428-011-0124-6

[34] R. Mccreadie, C. Macdonald, and I. Ounis, "Crowdsourcing a news query classification dataset," in *Proceedings of CSE*, 2010.

[35] A. M. McDonald and L. F. Cranor, "The cost of reading privacy policies," *Isjlp*, vol. 4, p. 543, 2008.

[36] M. Nakatani Shuyo, "langdetect: Language detection library ported from Google's language-detection." [Online]. Available: https://github.com/Mimino666/langdetect

[37] M. E. Peters and D. Lecocq, "Content extraction using diverse feature sets," in *Proceedings of the 22nd International Conference on World Wide Web - WWW '13 Companion*. Rio de Janeiro, Brazil: ACM Press, 2013, pp. 89–90. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2487788.2487828

[38] S. V. Rouse, "A reliability analysis of Mechanical Turk data," *Computers in Human Behavior*, vol. 43, pp. 304–307, Feb. 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0747563214005809

[39] N. Roy and A. McCallum, "Toward optimal active learning through sampling estimation of error reduction," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, p. 441–448.

[40] ——, "Toward optimal active learning through sampling estimation of error reduction," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, p. 441–448.

[41] F. Schaub, T. D. Breaux, and N. Sadeh, "Crowdsourcing the extraction of data practices from privacy policies," *Association for the Advancement of Artificial Intelligence (AAAI)*, p. 2, 2014.

[42] B. Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, pp. 1–114, Jun. 2012. [Online]. Available: http://www.morganclaypool.com/doi/abs/10.2200/S00429ED1V01Y201207AIM018

[43] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Honolulu, Hawaii: Association for Computational Linguistics, Oct. 2008, pp. 1070–1079. [Online]. Available: https://www.aclweb.org/anthology/D08-1112

[44] B. Settles, M. Craven, and L. Friedland, "Active learning with real annotation costs," *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, 01 2008.

[45] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, Jul. 1948. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6773024

[46] Y. Shen, H. Yun, Z. Lipton, Y. Kronrod, and A. Anandkumar, "Deep active learning for named entity recognition," in *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Vancouver, Canada: Association for Computational Linguistics, 2017, pp. 252–256. [Online]. Available: http://aclweb.org/anthology/W17-2630

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 06 2014.

[48] P. Story, S. Zimmeck, and N. Sadeh, "Which apps have privacy policies? - an analysis of over one million google play store apps," in *APF*, 2018.

[49] Y. P. Sun, "Investigating the effectiveness of android privacy policies," Master's thesis, University of Toronto, 2018.

[50] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for Twitter sentiment classification," in *Proceedings of the 52nd Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Baltimore, Maryland: Association for Computational Linguistics, 2014, pp. 1555–1565. [Online]. Available: http://aclweb.org/anthology/P14-1146

[51] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res.*, vol. 2, p. 45–66, Mar. 2002. [Online]. Available: https://doi.org/10.1162/153244302760185243

[52] T. Watanabe, M. Akiyama, T. Sakai, H. Washizaki, and T. Mori, "Understanding the inconsistency between behaviors and descriptions of mobile apps," *IEICE Transactions on Information and Systems*, vol. E101.D, no. 11, pp. 2584–2599, Nov. 2018. [Online]. Available: https://www.jstage.jst.go.jp/article/transinf/E101.D/11/E101.D_2017ICP0006/_article

[53] S. Wilson, F. Schaub, A. A. Dara, F. Liu, S. Cherivirala, P. Giovanni Leon, M. Schaarup Andersen, S. Zimmeck, K. M. Sathyendra, N. C. Russell, T. B. Norton, E. Hovy, J. Reidenberg, and N. Sadeh, "The creation and analysis of a website privacy policy corpus," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, 2016, pp. 1330–1340. [Online]. Available: http://aclweb.org/anthology/P16-1126

[54] S. Wilson, F. Schaub, R. Ramanath, N. Sadeh, F. Liu, N. A. Smith, and F. Liu, "Crowdsourcing annotations for websites' privacy policies: Can it really work?" in *Proceedings of the 25th International Conference on World Wide Web - WWW '16*. Montreal, Quebec, Canada: ACM Press, 2016, pp. 133–143. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2872427.2883035

[55] T. Winograd, "Procedures as a representation for data in a computer program for understanding natural language," 10 2004.

[56] L. Yang, Y. Zhang, J. Chen, S. Zhang, and D. Z. Chen, "Suggestive annotation: A deep active learning framework for biomedical image segmentation," 2017.

[57] Y. Zhang, M. Lease, and B. C. Wallace, "Active discriminative text representation learning," *arXiv:1606.04212 [cs]*, Dec. 2016, arXiv: 1606.04212. [Online]. Available: http://arxiv.org/abs/1606.04212

[58] L. Zhao, G. Sukthankar, and R. Sukthankar, "Incremental relabeling for active learning with noisy crowdsourced annotations," in *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, 2011, pp. 728–733.

[59] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 3485–3495. [Online]. Available: https://www.aclweb.org/anthology/C16-1329

[60] S. Zimmeck and S. M. Bellovin, "Privee: An architecture for automatically analyzing web privacy policies," in *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 1–16. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/zimmeck

[61] S. Zimmeck, P. Story, D. Smullen, A. Ravichander, Z. Wang, J. Reidenberg, N. Cameron Russell, and N. Sadeh, "MAPS: Scaling privacy compliance analysis to a million apps," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 66–86, Jul. 2019. [Online]. Available: https://content.sciendo.com/doi/10.2478/popets-2019-0037

[62] S. Zimmeck, Z. Wang, L. Zou, R. Iyengar, B. Liu, F. Schaub, S. Wilson, N. Sadeh, S. M. Bellovin, and J. Reidenberg, "Automated Analysis of Privacy Requirements for Mobile Apps," in *Proceedings 2017 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, 2017. [Online]. Available: https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/automated-analysis-privacy-requirements-mobile-apps/