An Empirical Study of Automated Privacy Requirements Classification in Issue Reports

Pattaraporn Sangaroonsilp[1*], Morakot Choetkiertikul[2], Hoa Khanh Dam[1] and Aditya Ghose[1]

[1*]School of Computing and Information Technology, Faculty of Engineering and Information Sciences, University of Wollongong, Northfields Avenue, Wollongong, 2500, NSW, Australia.
[2]Faculty of Information and Communication Technology, Mahidol University, Phuttamonthon Sai 4, Salaya, 73170, Nakhon Pathom, Thailand.

*Corresponding author(s). E-mail(s): ps642@uowmail.edu.au;
Contributing authors: morakot.cho@mahidol.ac.th;
hoa@uow.edu.au; aditya@uow.edu.au;

Abstract

The recent advent of data protection laws and regulations has emerged to protect privacy and personal information of individuals. As the cases of privacy breaches and vulnerabilities are rapidly increasing, people are aware and more concerned about their privacy. These bring a significant attention to software development teams to address privacy concerns in developing software applications. As today's software development adopts an agile, issue-driven approach, issues in an issue tracking system become a centralised pool that gathers new requirements, requests for modification and all the tasks of the software project. Hence, establishing an alignment between those issues and privacy requirements is an important step in developing privacy-aware software systems. This alignment also facilitates privacy compliance checking which may be required as an underlying part of regulations for organisations. However, manually establishing those alignments is labour intensive and time consuming. In this paper, we explore a wide range of machine learning and natural language processing techniques which can automatically classify privacy requirements in issue reports. We employ six popular techniques namely Bag-of-Words (BoW), N-gram Inverse Document Frequency (N-gram IDF), Term

Frequency-Inverse Document Frequency (TF-IDF), Word2Vec, Convolutional Neural Network (CNN) and Bidirectional Encoder Representations from Transformers (BERT) to perform the classification on privacy-related issue reports in Google Chrome and Moodle projects. The evaluation showed that BoW, N-gram IDF, TF-IDF and Word2Vec techniques are suitable for classifying privacy requirements in those issue reports. In addition, N-gram IDF is the best performer in both projects.

Keywords: Privacy, Issue reports, Issues classification, Machine Learning, Natural Language Processing, Deep learning models, Privacy issues classification

1 Introduction

In this digital era, software applications play an important role in providing a range of services for people in their daily lives (e.g. browsing information and using online learning platforms). People leave their digital footprints such as search history and personal details when interacting with those software applications. This raises critical concerns to their privacy and personal data protection. There are many data protection and privacy legislations and policies around the world put in place to govern personal data processing (e.g. General Data Protection Regulations (EU) [1] and California Consumer Privacy Act (US) [2]). These regulations provide a set of requirements for handling personal data in organisations. They also provide the rights for individuals to manage their personal data (e.g. right to be informed and right of access). The organisations needing to comply with these regulations are required to consider privacy compliance in their software systems. Failing to comply with these regulations may cause negative consequences to organisations in terms of reputation and financial hardship [3–6].

In addition, the cases of privacy breaches and vulnerabilities have been rapidly increasing. Those breaches not only occurred in small organisations but also happened to the world's top leading companies (e.g. Google and Marriott) [7–9]. These scenarios affect the organisations' reputation and exposed individuals. Hence, there is an urgent need to ensure that privacy and personal data protection are taken into consideration when developing software systems. It is however challenging for organisations to integrate privacy and personal data protection requirements into the existing processes, especially for deployed systems.

As an agile, issue-driven software development approach has been increasingly adopted in most today's software projects, issues become a main source of requirements of the software project [10]. Issues are lodged into issue tracking systems (e.g. JIRA) which are accessible for all the stakeholders who involve in the projects. For each development iteration, the development team selects a set of issues to work on for that iteration. Hence, the issue reports are the first source of requirements and project tasks that are considered by the

development team. Issues contain important information about new requirements (i.e. feature requests), change requests for existing requirements (i.e. improvements), reporting requirements not being properly met (i.e. bugs) or representing work that has to be done (i.e. tasks) [10, 11].

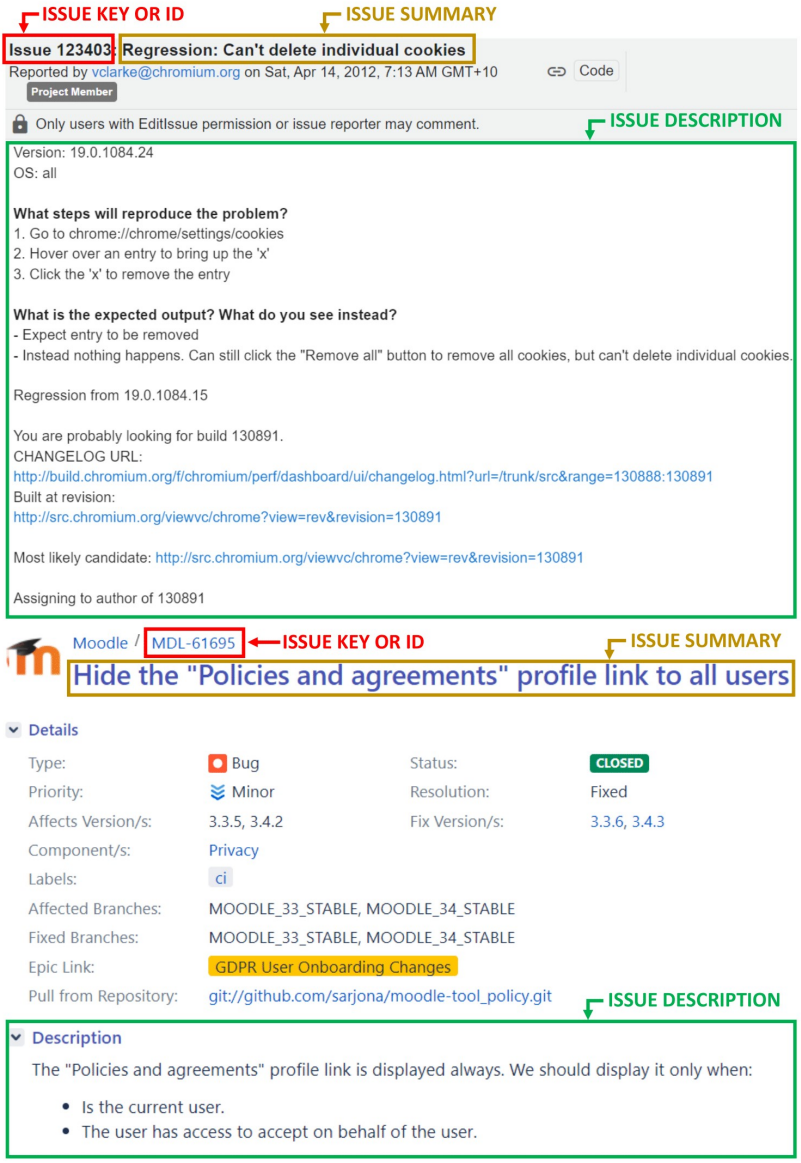

Fig. 1: An example of Google Chrome (top) and Moodle (bottom) issue reports.

Issue reports are normally written in *natural language* (see Figure 1). They contain information (e.g. issue key or ID, issue summary, issue description, issue type, priority, status and components) that describes scenarios and states actions needed to be attend by software engineers. Given thousands of issue reports in large software projects, it is challenging for software engineers to identify privacy-related issues. In addition, privacy concerns vary depending on functionalities and context provided by a software. For example, a web browser (e.g. Chrome) may prioritise the user search history while an online learning platform (e.g. Moodle) focuses on protecting user profiles and personal information. Hence, there is an emerging need to identify relevant privacy requirements in issue reports.

A recent study developed a taxonomy of privacy requirements from data protection regulations and privacy frameworks [12]. This taxonomy provides a set of fundamental privacy requirements for developing privacy-aware software applications. One of its usages is to classify privacy-related issues in a software project into relevant privacy requirements. This classification facilitates software development teams in identifying privacy requirements concerned in issue reports as well as ensuring that the associated privacy requirements are properly addressed. In addition, the process would also enable privacy compliance checking which requires a demonstration on the privacy needs and concerns in legislations are addressed in a software system. Since the taxonomy contains a large group of privacy requirements, the classification process is labour intensive and time consuming. Hence, an *automated support* is much needed for performing this task. The support could be provided in a "just-in-time" manner: once an issue is created or modified, the machinery (integrated with an issue-tracking system) will automatically classify the issue into appropriate privacy categories and requirements.

One prominent option to develop this automated solution is leveraging the usage of machine learning (ML) and natural language processing (NLP) techniques. Information in an issue report (e.g. title and description) is extracted into features. Those features are then used to build machine learners that are capable of learning from historical data to perform the classification on new data. There is a range of state-of-the-art techniques for extracting and learning those so-called textual features. In this study, we explore a wide range of machine learning and natural language processing techniques that can be used to automatically classify privacy requirements in issue reports. This paper provides the following contributions:

- We evaluate the performance of the traditional word embedding techniques (i.e. BoW [13], N-gram IDF [14], TF-IDF [15] and Word2Vec [16, 17]) and deep learning techniques (i.e. CNN [18] and BERT [19]) in classifying privacy requirements in issue reports. We use the labelled dataset published by Sangaroonsilp et al. [12] as input data in this empirical study. We employ *Mean Reciprocal Rank (MRR)* and *recall at k (Recall@k)* to compare the performance of each method. In addition,

we identify the best performing technique that can be used to assist a software team in identifying privacy requirements in issue reports. The results confirmed that N-gram IDF is the best performer with 0.6093 and 0.5838 on MRR in Google Chrome and Moodle projects respectively. TF-IDF also performs well on both MRR and recall@5 in both projects. It achieves 0.6093 on MRR in Google Chrome project and achieve the highest recall@5 at 0.7866 and 0.6027 in Google Chrome and Moodle respectively.

- We perform a Wilcoxon test to investigate if the classification results of all classification methods are statistically significant difference. We found that the recall@k results of random guessing method are statistically significant difference for all the traditional word embedding and deep learning techniques with effect sizes greater than 0.95 in Google Chrome project. In Moodle project, only the recall@k results between random guessing and BoW, N-gram IDF and TF-IDF, and between N-gram IDF and BERT are statistically significant difference with effect sizes greater 0.95.

A full replication package containing all the artefacts generated by our studies is publicly made available at [20]. The remainder of this paper is structured as follows. We introduce background and related work on a taxonomy for privacy requirements classification and text classification techniques in Section 2. In Section 3, we discuss on a motivating example and explain the approaches implemented in our study. The details of dataset, experimental setting, performance measures and evaluation results are presented in Section 4. We address threats to validity of our study in Section 5. Finally, we conclude and discuss future work in Section 6.

2 Background and Related Work

Personal data protection and privacy have attracted attention from individuals and organisations globally in recent years. After the announcement of GDPR in 2016, over hundred countries around the world have developed their own data protection and privacy legislations [21]. As people interact with software applications and systems, it is necessary to develop the software applications and systems that comply with those legislations to ensure personal data protection. However, it is challenging for software engineers to understand and implement privacy requirements when developing software applications in practice [22].

Several work identified privacy requirements and constructed privacy requirement taxonomies based on privacy policies and privacy standards [23, 24]. Antón and Earp [23] developed a privacy goal taxonomy from website privacy policies. The study adopted a Goal-based Requirement Analysis Method (GBRAM) to extract privacy goals and requirements from 24 online healthcare privacy policies. This taxonomy provides a useful set of requirements that the website developers could use to reduce web vulnerabilities.

Ayala-Rivera and Pasquale [24] proposed an approach to identify requirements that should be implemented in a software system. The requirements were elicited from the mapping between GDPR and privacy controls in ISO/IEC standards. Although both studies are relevant to privacy requirements elicitation and classification, none of them investigated privacy requirements in issue reports and studied on the automated classification process.

A number of existing studies used ML and NLP methods to assist in regulatory compliance checking [25, 26]. Müller et al. [25] studied on a method used to check the compliance between GDPR and companies' privacy policies. The statements in privacy policies were extracted and classified into five categories (i.e. Data Protection Officer (DPO), Purpose, Acquired data, Data sharing and Rights). The study applied three different word embedding techniques (i.e. Word2Vec, FastText [27] and ELMo [28]) in combination with three classifiers (i.e. Support Vector Machines (SVMs), Logistic Regression (LR) and Neural Networks (NN)) to automatically classify the extracted privacy policies' statements into those five categories. It employed F-measure to evaluate the classification performance. Similarly, Torre et al. [26] proposed a solution for completeness checking of privacy policies against GDPR. The study used a pre-trained word-vector model to transform sentences in privacy policies into vector representations [29]. It then built ML classifiers for classifying the sentences into metadata types. Precision and recall were used as performance measures. Comparing to our study, these studies focused on privacy policies, employed fewer word embedding techniques and classified data into a smaller group of categories.

A number of studies applied ML, NLP and deep learning models to perform issue report classification in different applications [10, 30–32]. Fan et al. [30] performed a large-scale study on issue reports of 80 popular projects in GitHub. They classified issues into a bug or non-bug type. The study evaluated the classification performance of four traditional text-based classifiers which are SVM, LR, Multinomial Naive Bayes (MNB) and Random Forest (RF). It used an average F-measure to evaluate the classification performance of those classifiers. It also proposed a new framework that can improve the performances of the traditional classification methods. Pandey et al. [31] studied various classification algorithms used to classify issue reports of three open-source software projects based on their types (i.e. bug or improvement). The algorithms consist of Naive Bayes (NB), linear discriminant analysis, k-nearest neighbours and SVM with various kernels, decision tree and RF. The authors employed F-measure, average accuracy and weighted average F-measure to evaluate the classification performance. The study reported that RF performed best in this setting.

Choetkiertikul et al. [10] proposed a predictive model that can be used to recommend the most relevant software components for new issue reports. The model was built using the deep learning Long Short-Term Memory (LSTM) technique. The issue reports from a collection of 11 open-source projects from four repositories were used as inputs for the model. The performance of the

model was evaluated using recall@k. Cho et al. [32] proposed a method that automatically classified issue reports into software feature descriptions in user manuals of three software projects: Notepad++, Visual Studio Code and Komodo. The authors performed an experiment to evaluate the classification performance of 8 approaches based on a combination of two deep learning models (i.e. CNN and Recurrent Neural Network (RNN)) and four word embedding techniques (i.e. embedding layer, Word2vec, GloVe and FastText). The study used precision, recall and F-measure to evaluate the performance of those approaches. The best performing word embedding techniques are FastText and Glove while CNN performed better than RNN in this experimental setting. Although the existing studies mentioned above investigated various classification problems in issue reports, they focused on types/components classification, not privacy-related issue reports. In addition, our study employed some different word embedding techniques, deep learning models and performance measures compared to those studies.

Recent work [12] developed a taxonomy of privacy requirements aiming to support the development of privacy-aware software systems. The work identified privacy requirements in issue reports and mapped them to relevant privacy requirements in the taxonomy. The process of taxonomy development consists of three main steps: privacy requirements identification, refinement and classification. In the privacy requirements identification step, the privacy requirements were derived from four well-established data protection regulations and privacy frameworks (i.e. General Data Protection Regulations (GDPR) [1], Thailand Personal Data Protection Act (PDPA) [33], ISO/IEC 29100 [34] and Asia-Pacific Economic Cooperation (APEC) privacy framework [35]).

GDPR is a data protection regulation developed by the European Union (EU). It provides a set of principles and conditions for protecting personal data and individual rights which need to be complied by applicable organisations. Thailand PDPA, based on GDPR, is a country-specific personal data protection regulation that governs the use and protection of personal data. ISO/IEC 29100 and APEC privacy framework define a set of privacy principles used to manage personal data processing activities. They also provide guidelines for controlling the processing of personal data. The processing of personal data includes collection, use, storage, dissemination and disposal.

The privacy requirements derived from multiple sources can be redundant and/or inconsistent, thus the privacy requirements refinement step was designed to remove duplicate requirements and manage inconsistent requirements. These privacy requirements were then classified into categories in the privacy requirements classification step. In this step, the authors identified the privacy requirements that have common characteristics or address similar privacy concerns, grouped those privacy requirements together, and finally formed the categories. This created a taxonomy of 71 privacy requirements with 7 categories which consist of (1) user participation, (2) notice, (3) user desirability, (4) data processing, (5) breach, (6) complaint/request and (7) security.

To validate the taxonomy, the authors of the paper performed a study on how issue reports of two large open-source software projects (Google Chrome and Moodle) address the privacy requirements in the taxonomy. Google Chrome and Moodle projects were selected due to their accessibility of issue reports, large scale size, popularity and representativeness. To identify and classify privacy requirements in issue reports, the authors first collected issue reports from issue tracking systems of Chrome and Moodle. There were 896 Chrome and 478 Moodle issue reports collected. Those issue reports were marked as *closed* and *privacy-related*.

Then, the issue classification process was performed by three coders, who were the authors of the paper and also involved in the taxonomy development process. Each issue report was annotated by two coders. The coders followed the following three steps to classify issue reports to relevant privacy requirements. First, the coders identified concerned personal data (e.g. name, email address and bank account details) described in an issue report. Then, they identified functions/properties related to the concerned personal data reported in the issue. Finally, they mapped the issue report to one or more relevant privacy requirements. The coder took approximately 138 person-hours to classify 1,374 issue reports. The coders also performed an inter-rater reliability assessment and a disagreement resolution to mitigate subjective judgements in issue report classification process. This process produced a dataset that contains Google Chrome and Moodle issue reports associated with concerned privacy requirements.

Although the work in [12] provides a valuable dataset for mining and classifying privacy requirements in issue reports, an *automated approach* has not been studied. As can be seen above, the issue classification process is labour intensive and time-consuming. Thus, we aim to automate this process by exploring and evaluating multiple feature extraction techniques, and reporting the best technique that can be used in this setting. The automated approach would reduce the use of resources and efforts as well as minimise human errors.

3 Classifying Privacy Requirements in Issue Reports

3.1 Motivating example

The following example demonstrates how the issue reports were manually classified into relevant privacy requirements in [12]. Issue 123403[1] in Google Chrome reported that users cannot delete individual cookies (see Figure 1). The coders read through the issue summary and description and followed the following three steps to classify issue reports to privacy requirements. First, the coders identified concerned personal data (e.g. name, email address and bank account details) described in the issue report. Then, they identified functions/properties related to the concerned personal data reported in the issue.

[1] https://bugs.chromium.org/p/chromium/issues/detail?id=123403

Finally, they mapped the issue report to one or more relevant privacy requirements. Based on this issue, the coders identified *cookies* as the concerned personal data as it may contain login details, search history and identifiable information. The coders identified *delete* as the function related to the cookies. Finally, this issue report was classified into requirement R44 ***ALLOW*** *the data subjects to erase their personal data* under Category 1 (User participation) in the taxonomy (see Appendix A). This scenario reflects that the browser should allow the users to select the cookies that they want to delete [12]. This manual classification process can be automated using ML and NLP techniques as well as deep learning models, which is studied in this work.

3.2 Approaches

Our study explores the feature extraction techniques that can be used to automatically classify privacy requirements in issue reports. We use the textual description (i.e. summary and description) of issue reports as an input to a model to classify relevant privacy requirements (see Figure 2). We treat our classification model as *multi-label multi-class classification problem* since an issue can be classified into multiple privacy requirements. We acquire the privacy-related issues from [12] (see Section 4.1 for more details). We first extract the summary and description of issues with the label(s) of relevant privacy requirements from the dataset. We then apply different textual feature extraction and learning techniques to form a vector representation of texts. Those vectors and privacy requirement labels are then fed to a classifier for training and validation.

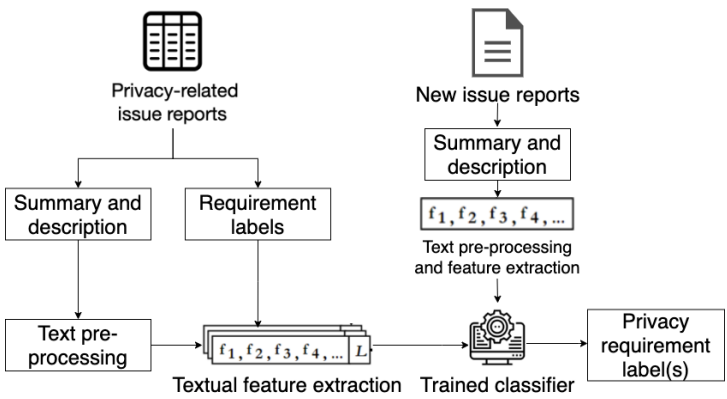

Fig. 2: An overview framework for developing a classification model.

The classification process involves three essential steps: textual data preprocessing, textual feature extraction, and training a classification model. The first important step is to eliminate noise from texts. In our study, we remove stop words from the input texts and also apply lemmatisation. Stop words are

the words that occur commonly in the texts but contain less meaning (e.g. a/an/the) while the lemmatisation reduces the variation of words by converting words into their root form (e.g. played to play). It is noted that the stop words removal and lemmatisation are not applied in BERT. The preprocessed texts are then encoded into vectors using a textual feature extraction technique. We describe each technique in Section 3.2.1 - 3.2.5 below. It is important to note that our models take as input the summary/title and description of an issue. This is the minimal information which must be provided at the time when an issue is created for any issue tracking system. Thus, the automated support provided by these classification models is readily available from the issue creation time. In addition, this enables our study to be applicable to a wide range of issue tracking systems.

The issue feature vectors derived from each method are then fed into a classifier. We first employed two well-known classifiers *RF and SVM*, that performed best in several document classification problems [36, 37]. However, we found that RF performed better than SVM for all the traditional word embedding techniques we adopted in our context. Thus, we decided to use only the RF classifier for all the traditional word embedding techniques in this study. RF is a randomised ensemble method where a classification is made by voting from weak learners (i.e. decision trees) [15]. We can thus derive the probability distribution among class labels as recommended requirements. We note that BERT has different details in the implementation steps. Hence, we specifically describe our BERT implementation in Section 3.2.6.

3.2.1 Bag of Words

Bag of Words (BoW) is the simplest model that represents texts as a vector of word frequency [13]. It basically creates a list of vocabulary for the entire corpus, then constructs a vector representation of a vocabulary size for each document. Elements in the vector contain the frequency of each word. BoW is implemented in our study as follows. After the issues are pre-processed, we first tokenise every issue in our dataset to convert from sentences into a collection of words (i.e. terms). We then build our dictionary (i.e. vocabulary) from a list of *unique* words. Next, we generate a vector of each issue by counting the occurrences of each word appearing in that issue based on the words that we have in our vocabulary set. Finally, we acquire a vector representation of a vocabulary size for each issue, whose elements contain occurrences of the words. The vectors are then fed to train the model.

However, BoW has two major weaknesses. Firstly, it does not capture the semantics of words. For example, it treats “Chrome”, “browser” and “school” equally, while semantically “Chrome” and “browser” are more related than “browser” and “school” or “Chrome” and “school”. Secondly, the order of the words in texts is ignored. For instance, BoW represents the same vector representation for these two different sentences “Chrome is good” and “Is Chrome good”.

3.2.2 N-gram IDF

N-gram IDF is an extension of IDF weighting scheme. It is developed to measure weights of phrases (i.e. terms of any length, where length is greater than 1) appeared in a collection of documents [14]. It gives more weights to the phrases (N-gram terms) that occur in fewer documents. We note that the N-gram library[2] is adopted in our implementation. N-gram helps us to distinguish the frequency of different phrases that contain the same set of words. For example, from "Chrome is good" and "Is Chrome good", we can detect their difference when we use 3-gram setting. When it combines with IDF, it can also identify significant phrases occurred in our issue reports.

In our implementation, we first construct a vocabulary set of N-gram terms. We specify the number of grams (i.e. words in a term) that we would like to extract to our dictionary. For example, an issue summary states "Modify cookies settings". If we create a dictionary using 1-gram, we get a list of "Modify", "cookies" and "settings" (3 terms). If we use 2-gram, we get a list of 2-word terms as "Modify cookies" and "cookies settings" (2 terms). Therefore, when we set a range of 1-word to 2-word terms, our vocabulary set contains 1-word and 2-word terms as "Modify", "cookies", "settings", "Modify cookies" and "cookies settings" (5 terms). We set the length of words from 1 to 10 in our experimental setting. This generates a set of 1-word to 10-word terms we have in our issue reports to include in the dictionary. We then find the occurrences of all the terms in each issue report. The weights of the N-gram terms are computed using the combination of IDF and Multiword Expression Distance (MED) in the space of information distance [14]. Finally, we create vectors to store the results obtained.

N-gram IDF suffers from computational time when processing large corpus. It also fails to process natural language texts that do not have spaces between words (e.g. Chinese and Japanese)

3.2.3 TF-IDF

TF-IDF is a type of frequency-based embedding where a vector representation of texts is computed from the term weighting measurement across documents in the corpus [15]. This technique evaluates whether a term is important to a document or in a collection of documents. The TF-IDF weight term-document is calculated from the product of two values - term frequency (tf) and inverse document frequency (idf) as shown below:

$$w_{t,d} = tf_{t,d} \times \log_{10}\left(\frac{N}{df_t}\right)$$

where $w_{t,d}$ is a tf-idf weighted term-document of term t in document d, $tf_{t,d}$ is a term frequency of term t in document d, N is a number of documents in a collection, and df_t is a number of documents in a collection that contains term t.

[2] https://github.com/iwnsew/ngweight

Comparing to our implementation, documents are issue reports and terms (t) are a list of words that we have in our dictionary. To construct a TF-IDF vector for an issue, we first create a dictionary from all the terms in our issue reports. We set the range to extract the terms from 1-word to 10-word terms in our setting. Next, we define N as the number of issue reports we have in each project. In each project, we count the number of times that term t appears in an issue, then divide by the total number of terms in the issue, and keep it as a term frequency. We then count the number of issue reports that contain the individual term in our dictionary, and keep it as document frequency of each term. Finally, we calculate the TF-IDF values of terms and store them in the elements in a vector representation of each issue report.

TF-IDF overcomes the major weaknesses of BoW. The common terms that frequently appear in issue reports but do not show their significance in other issue reports in the dataset are identified by their weights in vectors. TF-IDF also captures basic linguistic notions of terms (e.g. synonymy) [15]. However, semantic similarities between words and sequence of words in texts are not promised [38].

3.2.4 Word2Vec

Word2Vec is a two-layer neural networks that learns word features (e.g. the transitional probabilities between words). It then represents words as a group of numerical vectors in a vector space (i.e. word embedding) [16]. The dot product of two word vectors in the vector space reflects the similarity between these two words. Word2Vec is developed to tackle the following problems:

- Coverage: Co-occurrence words may not occur consecutively to each other. There are two model architectures proposed to solve this problem: Continuous Bag-of-Words (CBoW) [39] and Skip-gram [16].
- Space: The aforementioned techniques create vectors of vocabulary size. However, most of the elements inside each vector store zero. This could lead to sparse distribution.
- Speed: The computation is expensive when the vector space is large.

In our implementation, we employ Google's pretrained Word2Vec model[3] in our study. We first tokenise all the issue reports in the dataset and store it in a vocabulary set. Next, we generate one-hot encoding vectors for all the issue reports as inputs. Next, we feed the inputs into a Word2Vec architecture. The two layers of Word2Vec consist of hidden layer and output layer. The hidden layer is an encoder receiving input vectors while the output layer is a decoder storing word probabilities. In the training process, the input layer is the word embedding of each issue report while the output layer is privacy requirement labels of that issue report. The model performs the training by adjusting the weight and bias vector in the hidden layer from the given input and output vectors. We then use the hidden layer to perform privacy requirements classification on our test set. In the testing process, we feed the word embedding of each issue report in the test set to the model that we have trained (i.e. hidden

[3] https://code.google.com/archive/p/word2vec/

layer in the training process) to get the classification results. We have modified our CNN implementation to support a multi-label multi-class classification. We have changed an activation function from softmax to sigmoid, and had a dedicated output unit for each requirement (i.e. category)

3.2.5 Convolution Neural Network

Convolution Neural Network (CNN) is a deep learning architecture proposed in [18]. It consists of input layer, hidden layer (or convolutional layer) and output layer. We implement this method to support a multi-label multi-class classification task by using a sigmoid activation function and having a dedicated output unit for each requirement (i.e. category). We adopt Keras[4] and Google's pretrained Word2Vec model[5]. The pre-trained model is used to create a word embedding for each input vector. We use dropout (dropout rate is 0.5) to avoid overfitting [40]. Binary cross entropy, provided by Keras, is used as the network's loss function. For the setting of optimiser, we employ Adam with the learning rate adjustment techniques *(ReduceLROnPlateau)* to train the model. We connect the output layer which is privacy requirement labels to the model. We train the model after fitting all the settings.

3.2.6 BERT

In addition to the feature extraction techniques mentioned above, we also explore Bidirectional Encoder Representations from Transformers (BERT) as one of the candidates for performance comparison in classifying privacy requirements in issue reports. BERT is one of the latest breakthroughs in machine learning and NLP architecture. We use Huggingface Transformers library [41] and Tensorflow Keras API [42] in our implementation.

We have followed the process steps shown in Figure 3 for developing a recommendation model using BERT [19]. We concatenate issue summary and description of issue reports and use them as input texts. We then perform text pre-processing. In this step, we only remove non-alphanumeric characters from the input texts. We do not perform stop words removal and lemmatisation with BERT implementation since these processes will affect the context of original texts (e.g. negations and parts of speech). BERT tokeniser handles text preprocessing by itself [43, 44]. The dataset is then split into training and test set (see Section 4.2).

We employ BERT_{BASE} model which has 12 transformer blocks (i.e. encoders), 768 hidden layers and 12 attention heads. We load the BERT tokeniser and configure parameters[6] in the model (e.g. maximum length of tokens, regularisation, optimiser and classification metrics, batch size and learning epochs). We have calculated the average number of words in our input texts (113 words for Google Chrome and 90 words for Moodle). Hence, we set the maximum length of tokens at 120 for both projects as this number is

[4] https://keras.io/
[5] https://code.google.com/archive/p/word2vec/
[6] These parameters are shown in the source code included in the replication package [20].

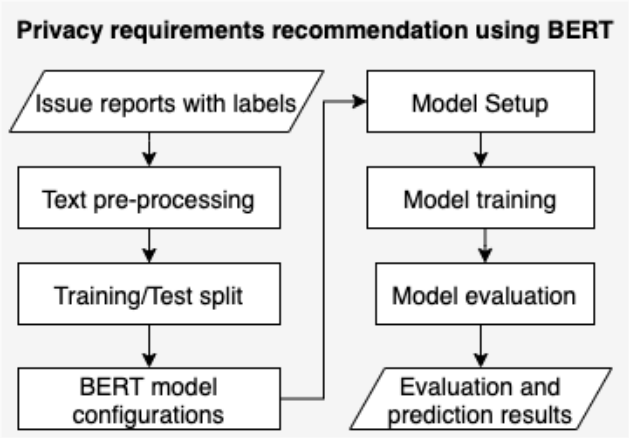

Fig. 3: An overview of privacy requirements classification process using BERT.

sufficient for our input. We denote that N represents the maximum length of token in Figure 4, thus N for our model is 120. After specifying all parameters, we perform the tokenisation process to transform input texts into tokens (represented as $Toki$ in Figure 4, where i is an order of token). In this step, the tokeniser also automatically adds a special token (i.e. the *[CLS]* token) into our original input text (see Figure 4). The *[CLS]* token is a starting point of each input text [19]. Finally, a complete sequence of input tokens is generated.

The next step is to set up our model input layer, main BERT layer and model outputs layer. From the previous step, each issue report is transformed into a sequence of input tokens. Thus, our model input layer is the sequences of input tokens. We load the BERT model with all configurations as a main BERT layer. Finally, we create an individual output for each privacy requirement (i.e. R1, R2, ... , R71). Since we perform multi-label multi-class classification, the model predicts a class label for individual privacy requirement of each issue report. We therefore have 71 model outputs.

Once the sequence of tokens is fed into the model, the model creates an input embedding (E_i) for each token as shown in Figure 4. The token embeddings are then forwarded to hidden states in each transformer block which functions as an encoder. BERT_{BASE} model consists of 12 transformer blocks, however Figure 4 shows only the first and last layers for illustration purpose. Next, the model is trained with the training set based on the setting above and evaluated on the validation set in the training process. We later evaluate the performance of our model on the test set.

4 Evaluation

We first describe the dataset used in this study and performance measures used for model evaluation. We then present and discuss the results obtained from privacy requirements classification in Google Chrome and Moodle projects.

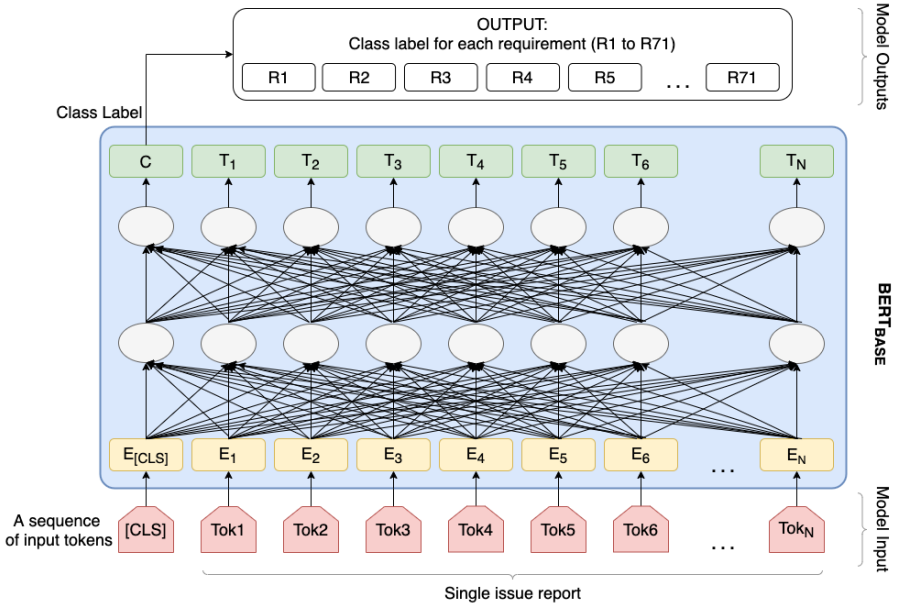

Fig. 4: An architecture of our BERT model (adapted from [19]).

4.1 Dataset

We use issue reports from two large and widely used software projects (i.e. Google Chrome and Moodle). This dataset[7] was originally collected and annotated by Sangaroonsilp et al. [12]. The authors first collected 1,604 issue reports in total from both projects. After the manual examination, 230 issues were filtered out due to limited information for classification task. Finally, the dataset contains 1,374 issues (896 issues from Google Chrome and 478 issues from Moodle). These issues were explicitly assigned *"privacy"* component in their issue tracking systems. In the dataset, each issue report contains issue ID, issue summary, issue description and its privacy requirement labels (see Figure 5).

As described in [12], an issue report was labelled with relevant privacy requirement(s). We use the term "labelled with" to denote the relationship between an issue report and its relevant privacy requirement(s). It is noted that some privacy requirements are belong to more than one category in the taxonomy. Each issue report can be represented in two levels: category and requirement levels. For example, issue 123403[8] in Google Chrome is classified into requirement R44 (see Figure 5a). Based on the taxonomy in [12], requirement R44 is categorised into category 1 (User participation) (see Figure 5b).

We also show the distribution of issue reports based on number of requirements they were classified to. The majority of the issues relates to one

[7]The dataset was publicly published at https://bit.ly/Mining-privacy-reqs-rev22
[8]https://bugs.chromium.org/p/chromium/issues/detail?id=123403

Issue ID	Issue Summary	Issue Description*	...	R42	R43	R44	R45	R46	...
123403	Regression: Can't delete individual cookies	Version: 19.0.1084.24 OS: all What steps will reproduce the problem? 1. Go to chrome://chrome/settings/cookies 2. Hover over an entry to bring up the 'x' 3. Click the 'x' to remove the entry ...	0	0	0	1	0	0	0

(a) An issue report with labels in requirement level.

Issue ID	Issue Summary	Issue Description*	Cat1	Cat2	Cat3	Cat4	Cat5	Cat6	Cat7
123403	Regression: Can't delete individual cookies	Version: 19.0.1084.24 OS: all What steps will reproduce the problem? 1. Go to chrome://chrome/settings/cookies 2. Hover over an entry to bring up the 'x' 3. Click the 'x' to remove the entry ...	1	0	0	0	0	0	0

(b) An issue report with labels in category level.

Fig. 5: An example of Google Chrome issue report with label in requirement and category levels. Irrelevant category/requirement to a particular issue report is represented by 0, whereas the relevant one is represented by 1. *Issue description is not shown in full details due to space limitation.

requirement for both projects (see Figure 6). Figure 7 shows the occurrences of all the privacy requirements in both projects. Out of 71 privacy requirements, the issue reports were classified into 39 and 40 requirements in Google Chrome and Moodle respectively. We have noted that our problem falls into a long tail distribution where the frequency of top requirements is heavy, which is challenging for the classification task.

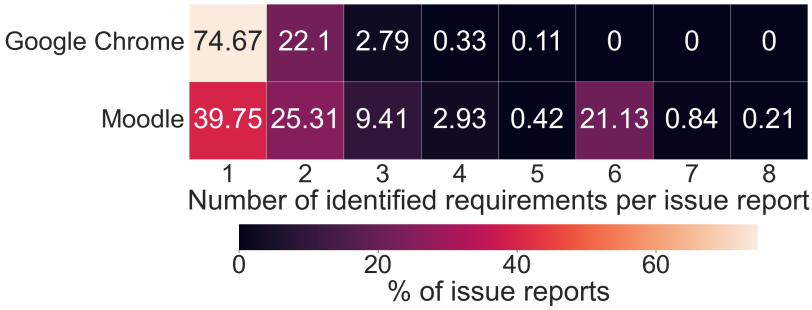

Fig. 6: The distribution of number of privacy requirements identified in Google Chrome and Moodle issue reports.

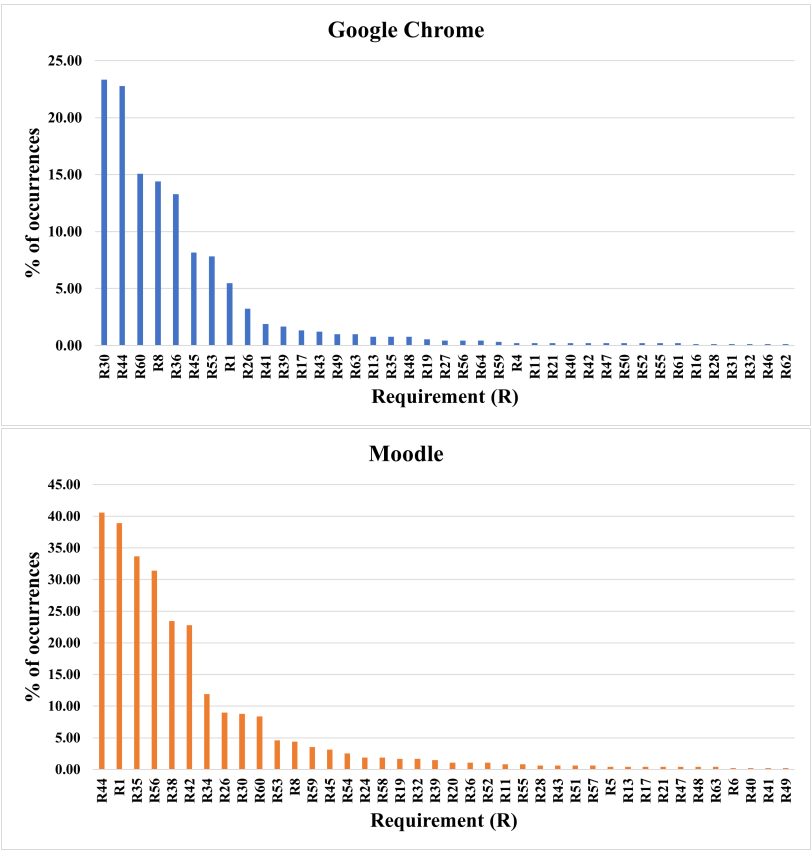

Fig. 7: Privacy requirements occurrences in Google Chrome and Moodle issue reports.

4.2 Experimental setting

We perform the experiments for each project separately. The issue reports are sorted by creation date based on their issue ID. The issues are then split into two mutually exclusive sets where those issues in training set are created before the issues in the test set to ensure that the models learn from historical data (60% of issues in the training set, 20% of issues in the validation set and 20% of issues in the test set).

4.3 Performance measures

We use two widely used measures (i.e. mean reciprocal rank (MRR) and recall at k (Recall@k)) to evaluate the performance of our models. We also employ a non-parametric hypothesis test (i.e. Wilcoxon rank-sum test) and effect size to compare classification benchmarks between a pair of models applied in our

study (see Table 2 and Table 3).

- MRR [45, 46] reflects the reciprocal of the rank at which the first relevant answer is recommended. It is calculated as:

$$MRR = \frac{1}{q}\sum_{i=1}^{q}\frac{1}{rank_i} \tag{1}$$

where q is the number of issue reports, $rank_i$ is the rank position of first relevant items.

For example, assume that we have two issue reports (i.e. A and B). Issue report A is labelled with requirement R5. The classifier returns requirements R1, R5 ,R12 for issue report A. Issue report B is labelled with requirements R7 and R3. The classifier returns requirement R7 for issue report B. The rank of issue report A is 2 as requirement R5 is in the second order from the list returned by the classifier, and the reciprocal rank is $\frac{1}{2}$. The rank of issue report B is 1 as requirement R7 is in the first rank of the returned result, and the reciprocal rank is 1. Hence, the MRR is:

$$MRR = \frac{1}{2}\left(\frac{1}{2}+1\right) = \frac{3}{4} \tag{2}$$

- Recall@k
Recall@k returns the number of correctly retrieved results over all the collected results from the top k elements recommended by our recommendation models. It is calculated as:

$$Recall@k = \frac{1}{k}\sum_{i=1}^{k}\frac{|Rec_i| \cap Label_i}{|Label_i|} \tag{3}$$

where k is a number of issue reports in the test set, Rec_i is a set of privacy requirements recommended for issue report i, and $Label_i$ is a set of privacy requirement(s) actually labelled in issue i.

Assuming that we have two issue reports (i.e. A and B), and three privacy requirements (i.e. R1, R2 and R3). Issue report A is actually labelled with requirements R1, R2, R3, while issue report B is actually labelled requirement R1. The classifier recommends requirements R1, R2 for issue report A, and requirements R1, R3 for issue report B. Assuming that we perform Recall@k, where $k = 2$, is:

$$\begin{aligned} Recall@2 &= \frac{1}{2}\left(\frac{|\{R1,R2\} \cap \{R1,R2,R3\}|}{|\{R1,R2,R3\}|} + \frac{|\{R1,R3\} \cap \{R1\}|}{|\{R1\}|}\right) \\ &= \frac{1}{2}\left(\frac{2}{3}+\frac{1}{1}\right) = \frac{5}{6} \end{aligned} \tag{4}$$

- Wilcoxon test
  Wilcoxon rank-sum test (also known as Mann-Whitney U test) is a non-parametric hypothesis test which compares the statistical difference between two independent observations [47]. We employ the Wilcoxon test to compare the significance of the performance of two classification models based on their recall@k, where k = 1 to k = 20.

  In addition, we also compute the common language effect size (CL) to reflect the proportion of paired observations where the first observation (x) is higher than the second observation (y) [48]. Later, Vargha and Delaney [49] proposed the generalisation of CL and called it as the *measure of stochastic superiority*. The measure of stochastic superiority ($\hat{A}_{XY}$ measure) is a generalisation of CL. This version does not require normally distributed data and it works with ordinal data. This measure was also recommended and employed in previous work to assess effect size when performing non-parametric hypothesis test [11, 50]. Therefore, it is suitable for our observations. ($\hat{A}_{XY}$ is calculated as:

$$\hat{A}_{XY} = \frac{[\#(X > Y) + (0.5 \times \#(X = Y))]}{n} \tag{5}$$

  where $\#(X > Y)$ is the number of observations that the recall@k of X is greater than the recall@k of Y, $\#(X = Y)$ is the number of observations that the recall@k of X equals Y, and n is the total number of observations (i.e. $n = 20$).

Table 1: Evaluation results of classifying privacy requirements in Google Chrome and Moodle issue reports.

Project	Text Feature	Classifier	MRR	Recall@k			
				k=5	k=10	k=15	k=20
Google Chrome	Baseline	Random	0.1693	0.1440	0.1694	0.3079	0.3588
		Freq.	0.4458	0.7477	0.9213	0.9523	0.9921
	BoW	RF	0.6049	0.7597	0.8977	0.9454	0.9509
	N-gram IDF	RF	**0.6093**	0.7866	0.9032	0.9245	0.9440
	TF-IDF	RF	**0.6093**	0.7866	0.9032	0.9245	0.9440
	Word2Vec	RF	0.5971	0.7755	0.8880	0.9338	0.9421
	CNN		0.4561	0.7477	0.9213	0.9523	0.9690
	BERT		0.5169	0.7097	0.9213	0.9644	0.9764
Moodle	Baseline	Random	0.2601	0.2624	0.3229	0.3486	0.4328
		Freq.	0.4111	0.4751	0.7222	0.8375	0.8615
	BoW	RF	0.5573	0.6027	0.7764	0.8186	0.8876
	N-gram IDF	RF	**0.5838**	0.5979	0.7392	0.8158	0.8674
	TF-IDF	RF	0.5573	0.6027	0.7764	0.8186	0.8876
	Word2Vec	RF	0.5431	0.5890	0.7632	0.8620	0.8765
	CNN		0.4126	0.4287	0.7222	0.8328	0.8696
	BERT		0.3898	0.4704	0.5934	0.8277	0.8538

4.4 Results

The methods we used in this work can be classified into 3 groups: naive baselines (i.e. random guessing and frequency-based method), traditional word embedding techniques (i.e. BoW, N-gram IDF, TF-IDF and Word2Vec) and deep learning techniques (i.e. CNN and BERT). We report here the results in answering the following research questions:

- **RQ1:** ***Do the traditional word embedding techniques outperform naive baselines?***

  This research question aims to examine how the selected traditional word embedding techniques perform compared to the naive baselines. We performed a sanity check by comparing the classification performance between the traditional word embedding techniques and two common naive baseline benchmarks, which are random guessing and frequency-based method. The random guessing performs random selection over a set of issues in the training set to give a recommendation based on the privacy requirements of the selected issue. The frequency-based method recommends privacy requirements based on the frequency of assigned privacy requirements in the training set. We note that the data used in this study is heavily imbalanced and has long tail distribution (see Figure 7). Thus, the frequency-based method becomes a strong baseline that the models should be compared with as a sanity check whether the techniques are suitable for classifying the privacy requirements. The goal of this RQ is to confirm that the selected traditional word embedding techniques outperform the naive baseline benchmarks and can be used to automatically classify privacy requirements in issue reports.

  Table 1 shows the MRR and recall@k results of all the state-of-the-art techniques and naive baseline benchmarks in Google Chrome and Moodle datasets. All the traditional word embedding techniques outperform both random guessing and frequency-based methods on MRR in both Google Chrome and Moodle projects. All the techniques except random guessing achieve over 0.7 recall@5 in Google Chrome project. In other words, more than 70% of the issue reports were classified into the correct privacy requirements when five requirements have been suggested. We note that as k equals to the number of privacy requirements that we have (i.e. 71), the recall will become 1 (i.e. issue reports are classified into every requirement). On recall@k, the traditional techniques also perform better than the random guessing baseline in both projects.

  In Google Chrome project, these techniques underperform the frequency-based baseline on average recall@k, where k = 1 to k = 20. N-gram IDF and TF-IDF are the best performers with 0.6093 on MRR in Google Chrome project. TF-IDF also achieves the highest recall@5. In Moodle project, the traditional techniques perform much better on MRR and recall@k. N-gram IDF achieves the highest MRR at 0.5838 while

BoW and TF-IDF achieve the highest recall@5 at 0.6027. All the traditional techniques perform much better than the random guessing and frequency-based methods. The differences on MRR and recall@k results between those four traditional techniques are very small (less than 0.05). In summary, N-gram IDF is the best performer in both projects.

- **RQ2:** ***Do deep learning techniques outperform the naive baselines and traditional word embedding techniques?***

  Since deep learning techniques have attracted a lot of attention from the research community and been using in various classification tasks [51], we aim to investigate if the selected deep learning techniques can beat the traditional word embedding techniques and naive baselines in issue report classification. We evaluated the classification performance of the selected deep learning techniques and compared them with those four traditional word embedding techniques and two naive baseline benchmarks. We found that CNN and BERT outperform the random guessing and frequency-based methods on MRR in both projects, except for BERT in Moodle project. Although they achieve higher recall@k, where k = 1 to k = 20, than the random guessing method, they cannot beat the frequency-based method in both projects.

  The deep learning techniques achieve lower MRR comparing to the traditional word embedding methods in both projects. CNN and BERT underperform those traditional methods on recall@k in Moodle project, but they outperform the traditional methods on recall@k in Google Chrome project. These two techniques may not be suitable for classifying issue reports into privacy requirements in this setting as the number of input data may not be sufficient for deep learning models.

- **RQ3:** ***Are the results statistically significant difference?***

  This RQ aims to investigate if the results of all classification methods are statistically significant difference. We can see that the results between naive baseline benchmarks, traditional word embedding techniques and deep learning techniques produced in the previous research questions are different, however we do not know how statistically significant difference those results are and how large of issue reports are concerned. Thus, we employed the Wilcoxon rank-sum test to answer this RQ.

  Table 2 and Table 3 show the results of the Wilcoxon test and their effect sizes evaluated on recall@k , where k = 1 to k = 20, in Google Chrome and Moodle respectively. In Google Chrome project, the Wilcoxon results confirm that there is a significant difference between the recall@k results of random guessing and all the other techniques. Those recall@k results are statistically significant difference with effect sizes greater than 0.95 for all the techniques in Google Chrome. The recall@k results of BERT are also statistically significant difference with

effect sizes equal to 0.98 for all four traditional word embedding techniques. The recall@k results among the traditional techniques are not statistically significant difference. In Moodle project, the recall@k results of all the techniques are not statistically significant difference, except the random guessing with BoW, N-gram IDF and TF-IDF and N-gram IDF with BERT. The effect sizes of those statistically different methods are greater than 0.95.

Based on our investigation in this study, we discuss here a few factors that should be considered and also provide some suggestions for researchers when classifying privacy requirements in issue reports. The first factor that we need to consider is a dataset. We have seen from our results that all the selected traditional word embedding techniques performed better than the selected deep learning-based techniques with the datasets we used. The deep learning-based methods, especially BERT, may suffer from small datasets. Another factor is the selection of feature extractors and classifiers. We suggest the researchers to perform a preliminary experiment to briefly assess the performance of privacy requirements classification in issue reports. The classification performance varies depending on the amount, quality and distribution of the data and experimental settings. In our setting, N-gram IDF with RF performs well in Google Chrome and Moodle datasets in this setting. The researchers can adopt this approach to see if it works well with their datasets.

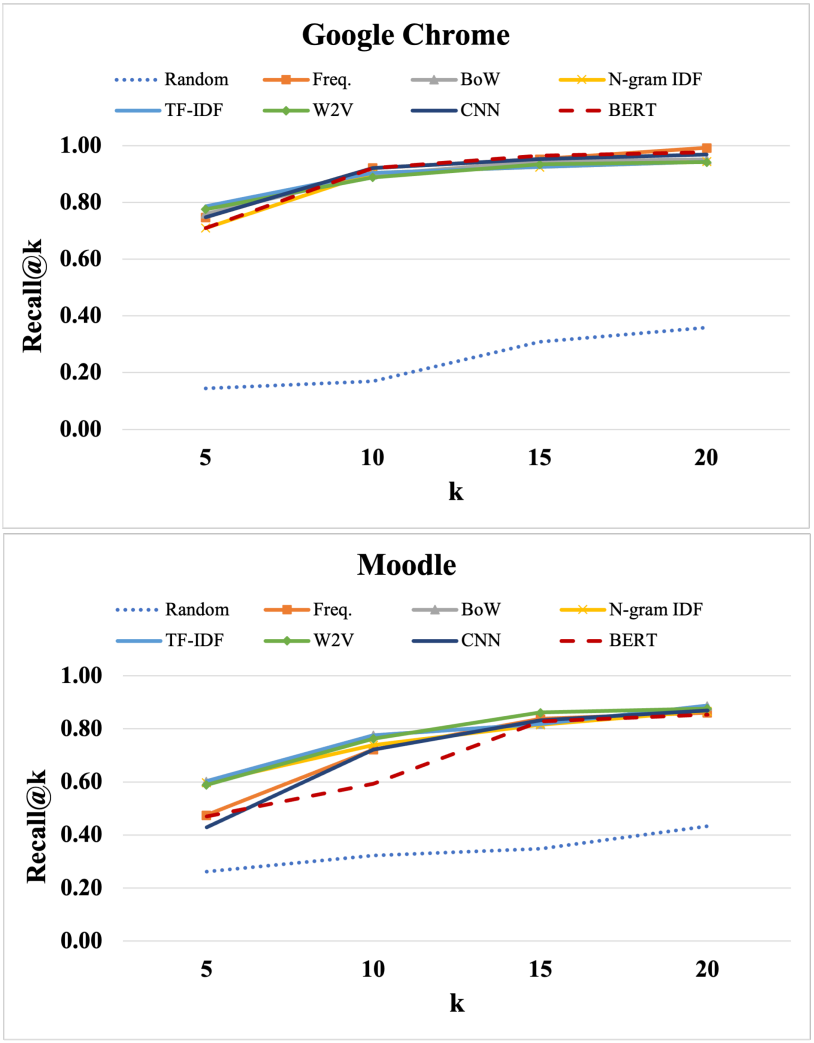

Fig. 8: The performance of textual feature extraction techniques employed in this study (evaluated on Recall@k).

Table 2: Comparison on classification benchmarks using Wilcoxon test and $\hat{A}_{XY}$ effect size (on the right column with 2 decimal points) on Google Chrome project.

vs	Random		Freq.		BoW		N-gram IDF		TF-IDF		Word2Vec		CNN		BERT	
Random			<0.001	0.03	<0.001	0.00	<0.001	0.00	<0.001	0.00	<0.001	0.00	<0.001	0.03	<0.001	0.05
Freq.	<0.001	0.97			0.465	0.57	0.298	0.60	0.298	0.60	0.440	0.57	0.914	0.51	0.000	0.94
BoW	<0.001	1.00	0.465	0.43			0.626	0.55	0.626	0.55	0.579	0.55	0.401	0.42	<0.001	0.98
N-gram IDF	<0.001	1.00	0.298	0.40	0.626	0.45			1.000	0.50	0.828	0.48	0.256	0.39	<0.001	0.98
TF-IDF	<0.001	1.00	0.298	0.40	0.626	0.45	1.000	0.50			0.829	0.48	0.256	0.39	<0.001	0.98
Word2Vec	<0.001	1.00	0.440	0.43	0.579	0.45	0.829	0.52	0.829	0.52			0.336	0.41	<0.001	0.98
CNN	<0.001	0.97	0.914	0.49	0.401	0.58	0.256	0.61	0.256	0.61	0.336	0.59			0.000	0.94
BERT	<0.001	0.96	0	0.06	<0.001	0.02	<0.001	0.02	<0.001	0.02	<0.001	0.02	0.000	0.06		

Table 3: Comparison on classification benchmarks using Wilcoxon test and $\hat{A}_{XY}$ effect size (on the right column with 2 decimal points) on Moodle project.

vs	Random		Freq.		BoW		N-gram IDF		TF-IDF		Word2Vec		CNN		BERT	
Random			0.000	0.09	<0.001	0.04	<0.001	0.03	<0.001	0.04	0.000	0.08	0.000	0.13	0.946	0.50
Freq.	0.000	0.91			0.499	0.44	0.882	0.49	0.499	0.44	0.273	0.40	0.946	0.51	0.000	0.91
BoW	<0.001	0.96	0.499	0.56			0.543	0.56	1.000	0.50	0.989	0.50	0.490	0.57	0.000	0.95
N-gram IDF	<0.001	0.98	0.882	0.52	0.543	0.44			0.543	0.44	0.579	0.45	0.839	0.52	<0.001	0.97
TF-IDF	<0.001	0.96	0.499	0.56	1.000	0.50	0.543	0.56			0.989	0.50	0.490	0.57	0.000	0.95
Word2Vec	0.000	0.92	0.273	0.60	0.989	0.50	0.579	0.55	0.989	0.50			0.379	0.58	0.000	0.92
CNN	0.000	0.87	0.946	0.49	0.490	0.44	0.839	0.48	0.490	0.44	0.379	0.42			0.000	0.88
BERT	0.946	0.51	0.000	0.10	0.000	0.05	<0.001	0.03	0.000	0.05	0.000	0.08	0.000	0.12		

5 Threats to validity

There are several threats to validity of our study which are discussed below.

5.1 Construct validity

We performed the privacy requirements classification in issue reports of Google Chrome and Moodle projects. The dataset used in our study was publicly published and have been used by previous work [12]. The issue reports in that dataset have been extracted from the active issue tracking systems of two large and widely-used software systems. Both projects have a strong emphasis on privacy concerns. We however acknowledge that it is a threat to construct validity as we rely on the labelled dataset of the previous study. Since the dataset involved with subjective judgements and was manually annotated, it may contain errors caused by human biases.

5.2 Internal validity

We are also aware that the configurations of parameters could affect the performance of the techniques applied in the experiments. We tried to minimise this threat by setting the same values for all the common parameters across different techniques in both Google Chrome and Moodle projects. In addition, our dataset has long tail distributions which may flavour the performance of the frequency-based method. However, we have used a range of different text features to perform the experiments and compare their performance. The results show that several text features performed better than the frequency-based method.

5.3 External validity

We acknowledge that our dataset may not be representative of other software applications or software applications in other domains. Further investigation is required to explore other projects in different domains (e.g. e-health software systems and mobile applications), which will be explored in our future work.

6 Conclusion and Future Work

User privacy in software development has attracted attention from the software development community in the past recent years. Together with the enactment of data protection regulations and laws, it is essential for software development teams to properly address privacy concerns in their software systems. In this paper, we have explored a wide range of textual feature techniques that can be used to automatically classify privacy requirements in issue reports. We performed our study on issue reports of Google Chrome and Moodle. We used MRR and recall@k to evaluate the performance of these techniques. The evaluation results showed that BoW, N-gram IDF, TF-IDF, Word2Vec are suitable for privacy requirements classification. In addition, N-gram IDF is

the best performer in our experiment. We believe that this study provides insightful reference in choosing textual feature extraction techniques in future work. Our future work involves studying on the robustness performance and in-depth analysis of different textual feature extraction techniques. We plan to extend our study on privacy-related issue reports of other software projects (e.g. health and mobile applications). We also plan to explore this line of research further and develop it as an automated tool to support the privacy requirements identification in issue tracking systems (e.g. JIRA).

Appendix A

A taxonomy of privacy requirements developed by Sangaroonsilp et al. [12] is listed below.

Category 1: User Participation
R1 ALLOW the data subjects to access and review their personal data
R2 ALLOW the data subjects to lodge a complaint with a supervisory authority
R3 ALLOW the data subjects to object to the processing of their personal data
R4 ALLOW the data subjects to restrict the processing of their personal data
R5 ALLOW the data subjects to verify the validity and correctness of their personal data
R6 ALLOW the data subjects to withdraw consent
R34 ALLOW the data subjects to obtain and reuse their personal data for their own purposes across different services
R44 ALLOW the data subjects to erase their personal data
R45 ALLOW the data subjects to rectify their personal data

Category 2: Notice
R9 INFORM the data subjects about the transfer of their personal data to a third country or an international organisation
R10 INFORM the data subjects prior to the elimination of their restriction of processing
R11 INFORM the data subjects the individual rights relating to the processing of personal data
R12 INFORM the data subjects the reason(s) for not taking action on their request and the possibility of lodging a complaint
R14 NOTIFY the data subjects if their personal data will be processing on other legal basis
R15 NOTIFY the data subjects the data breach which is likely to result in high risk
R16 NOTIFY the data subjects when major changes in the personal data handling procedures occur
R17 SHOW the relevant stakeholders the consent given by the data subjects to process their personal data
R18 PROVIDE the data subjects an option not to be subject to a decision solely based on automated processing
R19 PROVIDE the data subjects an option to choose whether or not to provide their personal data
R20 PROVIDE the data subjects with the contact details of a data protection officer (DPO)
R21 PROVIDE the data subjects the existence and relevant information of automated decision-making including profiling
R22 PROVIDE the data subjects with the identity and contact details of a

controller/controller's representative
R23 PROVIDE the data subjects the information about the source of personal data if they are not directly provided by himself/herself
R24 PROVIDE the data subjects the information of action taken on their request of individual's rights
R25 PROVIDE the data subjects the information of granting or withholding consent
R26 PROVIDE the data subjects the information relating to the policies, procedures, practices and logic of the processing of personal data
R27 PROVIDE the data subjects the recipients/categories of recipients of their personal data
R28 PROVIDE the data subjects the sufficient explanation for the need to process sensitive personal data
R29 PROVIDE the data subjects the consequences of not providing personal data based on the statutory or contractual requirement
R30 PROVIDE the data subjects the information relating to the processing of personal data with standardised icons
R37 PROVIDE the data subjects the additional information when further processing is required for a purpose other than the consent obtained
R38 PROVIDE the data subjects the purpose(s) of the collection of personal data
R39 PROVIDE the data subjects the purpose(s) of the processing of personal data
R42 PROVIDE the data subjects the categories of personal data concerned
R50 INFORM the recipients of personal data any rectification or erasure of personal data or restriction of processing
R54 PROVIDE the data subjects the data retention and disposal requirements
R55 PROVIDE the data subjects the period/criteria used to store their data
R66 NOTIFY a supervisory authority the data breach
R67 NOTIFY relevant privacy stakeholders about a data breach
R68 PROVIDE the data processor a channel to notify a data breach
R71 PROVIDE a supervisory authority the information about a data breach

Category 3: User Desirability
R6 ALLOW the data subjects to withdraw consent
R8 IMPLEMENT the data subject's preferences as expressed in his/her consent
R18 PROVIDE the data subjects an option not to be subject to a decision solely based on automated processing
R19 PROVIDE the data subjects an option to choose whether or not to provide their personal data
R25 PROVIDE the data subjects the information of granting or withholding consent
R32 SHOW the data subjects a consent form in an intelligible and easily accessible form using clear and plain language

R35 OBTAIN the opt-in consent for the processing of personal data for specific purposes
R36 PRESENT the data subjects an option whether or not to allow the processing of personal data
R47 ERASE the personal data when a consent is withdrawn

Category 4: Data Processing
R7 ERASE the personal data when it has been unlawfully processed
R9 INFORM the data subjects about the transfer of their personal data to a third country or an international organisation
R13 MAINTAIN a record of personal data processing activities
R33 TRANSMIT the personal data to another controller when requested by the data subjects
R46 ERASE the personal data when the data subjects object to the processing
R40 USE the personal data as necessary for specific purposes specified by the controller
R41 COLLECT the personal data as necessary for specific purposes
R43 STORE the personal data as necessary for specific purposes
R47 ERASE the personal data when a consent is withdrawn
R49 IMPLEMENT the personal data collection procedures to ensure with accuracy and quality
R51 ARCHIVE the personal data when required by laws
R52 ERASE the personal data when it is no longer necessary for the specified purpose(s)
R53 ERASE the personal data when the purpose for the processing has expired
R59 IMPLEMENT appropriate technical and organisational measures to ensure the personal data is collected, processed, and stored as necessary
R69 DOCUMENT the details of data breach to a supervisory authority for compliance verification
R70 DOCUMENT the categories of personal data collected

Category 5: Breach
R15 NOTIFY the data subjects the data breach which is likely to result in high risk
R66 NOTIFY a supervisory authority the data breach
R67 NOTIFY relevant privacy stakeholders about a data breach
R68 PROVIDE the data processor a channel to notify a data breach
R69 DOCUMENT the details of data breach to a supervisory authority for compliance verification
R71 PROVIDE a supervisory authority the information about a data breach

Category 6: Complaint/Request
R2 ALLOW the data subjects to lodge a complaint with a supervisory authority

R24 PROVIDE the data subjects the information of action taken on their request of individual's rights
R12 INFORM the data subjects the reason(s) for not taking action on their request and the possibility of lodging a complaint
R31 REQUEST the data subjects the additional information necessary to confirm their identity when making a request relating to the processing of personal data
R33 TRANSMIT the personal data to another controller when requested by the data subjects

Category 7: Security
R48 IMPLEMENT control mechanisms to regularly check the accuracy and quality of collected and stored personal data
R49 IMPLEMENT the personal data collection procedures to ensure with accuracy and quality
R56 ALLOW the authorised stakeholders to access personal data as instructed by a controller R57 IMPLEMENT a process for regularly assessing the effectiveness of the measures to ensure the security of processing
R58 IMPLEMENT appropriate technical and organisational measures to comply with data protection principles
R59 IMPLEMENT appropriate technical and organisational measures to ensure the personal data is collected, processed, and stored as necessary
R60 IMPLEMENT appropriate technical and organisational measures to protect personal data
R61 IMPLEMENT the ability to ensure the ongoing security principles and resilience of processing systems and services
R62 IMPLEMENT the ability to restore availability and access to personal data in timely manner after physical or technical incidents
R63 PROTECT the personal data from unauthorised access and processing
R64 IMPLEMENT a function to limit the linkability of personal data collected
R65 IMPLEMENT a function to comply with local requirements and cross-border transfers
R69 DOCUMENT the details of data breach to a supervisory authority for compliance verification

References

[1] Office Journal of the European Union: General Data Protection Regulation (GDPR). Technical report (2016). https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

[2] State of California Department of Justice: California Consumer Privacy Act (CCPA) (2018). https://oag.ca.gov/privacy/ccpa

[3] Data Privacy Manager: 5 biggest GDPR fines so far [2020] – Data Privacy Manager (2020). https://dataprivacymanager.net/5-biggest-gdpr-fines-so-far-2020/ Accessed 2020-03-24

[4] European Commission: GDPR IN NUMBERS 2019 #HAPPYBIRTHDAYGDPR. Technical report (2019)

[5] CNET: British Airways faces $230M GDPR fine for 2018 data breach (2019). https://cnet.co/2EHboiu

[6] Privacy Affairs: GDPR Fines List: Find all GDPR fines & detailed statistics (2020). https://www.privacyaffairs.com/gdpr-fines/

[7] BBC: British Airways fined £20m over data breach (2020). https://www.bbc.com/news/technology-54568784

[8] Reuters: ICO fines Marriott 18.4 million pounds for failing to secure customer data (2020). https://www.reuters.com/article/us-marriott-intnl-ico-idUSKBN27F1LH

[9] Swinhoe, D.: The 14 biggest data breaches of the 21st century (2020). https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html Accessed 2020-03-29

[10] Choetkiertikul, M., Dam, H.K., Tran, T., Pham, T., Ragkhitwetsagul, C., Ghose, A.: Automatically recommending components for issue reports using deep learning. Empirical Software Engineering **26**(2) (2021) https://doi.org/10.1007/s10664-020-09898-5

[11] Choetkiertikul, M., Dam, H.K., Tran, T., Pham, T., Ghose, A.: Poster: Predicting components for issue reports using deep learning with information retrieval. Proceedings - International Conference on Software Engineering, 244–245 (2018)

[12] Sangaroonsilp, P., Dam, H.K., Choetkiertikul, M., Ragkhitwetsagul, C., Ghose, A.: A Taxonomy for Mining and Classifying Privacy Requirements in Issue Reports. Information and Software Technology **157** (2023) https://doi.org/10.1016/j.infsof.2023.107162

[13] Tirilly, P., Claveau, V., Gros, P.: Language modeling for bag-of-visual words image categorization. In: Proceedings of the International Conference on Content-based Image and Video Retrieval, pp. 249–258 (2008)

[14] Shirakawa, M., Hara, T., Nishio, S.: N-gram IDF: A global term weighting scheme based on information distance. In: Proceedings of the 24th International Conference on World Wide Web, pp. 960–970 (2015)

[15] Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. The Journal of Machine Learning Research **3**, 993–1022 (2003)

[16] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: 1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings (2013)

[17] Google: word2vec (2013). https://code.google.com/archive/p/word2vec/ Accessed 2023-02-16

[18] Lee, S.R., Heo, M.J., Lee, C.G., Kim, M., Jeong, G.: Applying deep learning based automatic bug triager to industrial projects. In: Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering, vol. Part F1301, pp. 926–931 (2017). https://doi.org/10.1145/3106237.3117776

[19] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding (2018). https://arxiv.org/pdf/1810.04805.pdf

[20] Sangaroonsilp, P., Choetkiertikul, M., Dam, H.K., Ghose, A.: Replication Package for Automated Classification of Privacy Requirements in Issue Reports (2022). http://bit.ly/AutomaticPRrecommendation

[21] United Nations Conference on Trade and Development (UNCTAD): Data Protection and Privacy Legislation Worldwide (2020). https://unctad.org/page/data-protection-and-privacy-legislation-worldwide Accessed 2021-05-19

[22] Gürses, S., Troncoso, C., Diaz, C.: Engineering Privacy by Design. In: Computers, Privacy & Data Protection, pp. 1–25 (2011)

[23] Antón, A.I., Earp, J.B.: A requirements taxonomy for reducing Web site privacy vulnerabilities. Requirements Engineering **9**(3), 169–185 (2004)

[24] Ayala-Rivera, V., Pasquale, L.: The grace period has ended: An approach to operationalize GDPR requirements. In: Proceedings - 2018 IEEE 26th

International Requirements Engineering Conference, RE 2018, pp. 136–146 (2018)

[25] Müller, N.M., Kowatsch, D., Debus, P., Mirdita, D., Böttinger, K.: On GDPR Compliance of Companies' Privacy Policies. In: Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) vol. 11697, pp. 151–159 (2019)

[26] Torre, D., Abualhaija, S., Sabetzadeh, M., Briand, L., Baetens, K., Goes, P., Forastier, S.: An AI-assisted Approach for Checking the Completeness of Privacy Policies Against GDPR. 2020 IEEE 28th International Requirements Engineering Conference (RE), 136–146 (2020)

[27] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017) https://doi.org/10.1162/tacl_a_00051 https://arxiv.org/abs/1607.04606

[28] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, vol. 1, pp. 2227–2237 (2018). https://doi.org/10.18653/v1/n18-1202 . https://arxiv.org/abs/1802.05365v2

[29] Pennington, J., Socher, R., Manning, C.D.: GloVe: Global vectors for word representation. In: EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, pp. 1532–1543 (2014). https://doi.org/10.3115/v1/d14-1162 . https://nlp.stanford.edu/projects/glove/

[30] Fan, Q., Yu, Y., Yin, G., Wang, T., Wang, H.: Where Is the Road for Issue Reports Classification Based on Text Mining? In: International Symposium on Empirical Software Engineering and Measurement, vol. 2017-Novem, pp. 121–130 (2017). https://doi.org/10.1109/ESEM.2017.19

[31] Pandey, N., Sanyal, D.K., Hudait, A., Sen, A.: Automated classification of software issue reports using machine learning techniques: an empirical study. Innovations in Systems and Software Engineering **13**(4), 279–297 (2017) https://doi.org/10.1007/s11334-017-0294-1

[32] Cho, H., Lee, S., Kang, S.: Classifying issue reports according to feature descriptions in a user manual based on a deep learning model. Information and Software Technology **142**, 106743 (2022) https://doi.org/10.1016/j.infsof.2021.106743

[33] National Legislative Assembly: The Thailand Personal Data Protection Act. Technical report (2019). https://thainetizen.org/wp-content/uploads/2019/11/thailand-personal-data-protection-act-2019-en.pdf

[34] ISO/IEC: International Standard ISO/IEC 29100. Technical report, ISO/IEC (2011)

[35] Asia-Pacific Economic Cooperation (APEC): APEC Privacy Framework (2015). Technical report, APEC Secretariat (2017)

[36] Moraes, R., Valiati, J.F., Gaviao Neto, W.P.: Document-level sentiment classification: An empirical comparison between SVM and ANN. Expert Systems with Applications **40**(2), 621–633 (2013)

[37] Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)

[38] Kowsari, K., Meimandi, K.J., Heidarysafa, M., Mendu, S., Barnes, L., Brown, D.: Text classification algorithms: A survey. In: Information, vol. 10, pp. 1–68 (2019). https://www.mdpi.com/2078-2489/10/4/150

[39] Harris, Z.S.: Distributional Structure. WORD **10**(2-3), 146–162 (1954)

[40] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**, 1929–1958 (2014)

[41] Huggingface: Transformers (2020). https://huggingface.co/transformers/

[42] TensorFlow: The Sequential Model (2020). https://www.tensorflow.org/guide/keras/sequential{_}model

[43] Huggingface: Tokenizers (2020). https://github.com/huggingface/tokenizers Accessed 2021-01-10

[44] Reina, Y.: Text preprocessing for BERT (2020). https://www.kaggle.com/c/google-quest-challenge/discussion/127881

[45] Almhana, R., Mkaouer, W., Kessentini, M., Ouni, A.: Recommending relevant classes for bug reports using multi-objective search. In: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 286–295 (2016)

[46] Ye, X., Fang, F., Wu, J., Bunescu, R., Liu, C.: Bug Report Classification Using LSTM Architecture for More Accurate Software Defect Locating. In: Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018, pp. 1438–1445 (2019)

[47] Wild, C.: The Wilcoxon Rank-Sum Test. Technical report, Department

of Statistics, University of Auckland (1997)

[48] McGraw, K.O., Wong, S.P.: A Common Language Effect Size Statistic. Psychological Bulletin **111**(2), 361–365 (1992)

[49] Vargha, A., Delaney, H.D.: A critique and improvement of the CL common language effect size statistics of McGraw and Wong. Journal of Educational and Behavioral Statistics **25**(2), 101–132 (2000)

[50] Arcuri, A., Briand, L.: A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. Software Testing Verification and Reliability **24**(3), 219–250 (2014)

[51] Sarker, I.H.: Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. Springer (2021)