Extracting User Requirements from Social Media Using Machine Learning

Tahira Iqbal

fortiss — An-Institut Technical University Munich, Germany
iqbal@fortiss.org

1 Introduction

Determining and managing users needs related to software and hardware is known as Requirement Engineering (RE). RE is a key phase in the software development. In the past, RE was considered only in the initial phase of software life cycle, e.g. waterfall model, and it was not integrated into the remaining software development life cycle. As the time passed, software development models evolved, and it became part of complete software development life cycle. Software systems are developed over millions of lines of code, a number of modules and documents. The primary goal of the software system is to satisfy users by developing the software that can meet their needs and expectations. This goal is achievable by applying different methodologies and engineering techniques. One of the important factors is to understand and identify the needs of users, also known as, software requirements. Software requirement engineering is the process that helps to determine the requirements systematically to know what functionalities the targeted system should have to fulfill users needs. Formally RE is defined as [1] :
" Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families."

Software requirements plays a key role in the success of a project. In the USA, a survey was conducted over 8380 projects by 350 companies to know the project failure rates. The report [2] results showed only 16.2% projects were completed successfully and one-half (52.7%) of projects met with challenges and were completed with partial functionalities, time delays and over budget. Almost 31% of the projects were never completed. The main cause told by the executive managers was poor requirements engineering. The major problems were the lack of user involvement (13%), requirements incompleteness (12%), changing requirements (11%), unrealistic expectations (6%), and unclear objectives (5%).

Software requirement engineering has mainly four phases; requirement elicitation, requirement analysis, requirement documentation and requirement verification[3]. Requirement elicitation [4] [5] helps to understand the stakeholder's needs, e.g. what features he wants in the software. Re-

quirement elicitation techniques are mostly derived from social sciences, organizational theory, knowledge engineering and practical experience. For requirements elicitation, different techniques exist in the literature that include interviews, questionnaires, and ethnography etc. Requirement analysis [6] is the next step after requirement elicitation. In this phase, software requirements are analyzed to check conflicts and consistency. It also makes sure that the requirements are clear, complete and consistent. Furthermore, the agreed requirements are documented. This documentation has a clear and precise definition of the system functionalities. It also acts as an agreement between stakeholders and developers. These functionalities and requirements are documented usually as diagrams, mathematical formulae or natural languages. These documents are used and iterated until the end of the project.

System requirements are classified into functional requirements (FR), non-functional requirements (NFR), user requirements, business requirements. Functional requirements are the system requirements that include the main features and characteristics of the desired system. Non-functional requirements are the system properties and constraints [7] [8]. NFRs set the criteria for judging the operation of the system such as performance, availability, and reliability etc. Business requirements are specified to address business objective, vision, and goal. It is defined at a higher level from organization or company and considered for designing the products. User requirements are the wish list for the system from users. User requirements are valuable for ensuring the system performs similar to user needs.

1.1 Machine Learning

Machine learning (ML) is an emerging field of this era and is a part of Artificial Intelligence (AI). It helped to solve complicated and hard problems in software engineering efficiently [9]. ML mainly relies on the data and its algorithms learn from existing datasets. It also helps to predict solutions for unseen problems. Automated learning processes took great inspiration from human learning i.e. learning from examples. ML algorithms have proved to have a significant impact in different fields e.g. business, medical, software engineering, computer security, data and communication networks and others. For the leaning process algorithm, another key factor is feature. Features are the selected properties of a problem that help in predicting the results. As an example, for a disease diagnostic

software, patient symptoms are the features. List of symptoms is mapped to the particular disease. Now, if a new patient comes with some symptoms, the system would be able to predict the disease by learning from the existing dataset. In addition, features help to abstract the complexities of the information provided for learning. In short, a good selection of features leads to better and faster learning.

ML algorithms are divided into two major categories; supervised learning and unsupervised learning [7] [10]. A third category is reinforcement learning which has been added at a later stage. In supervised learning, data set is labeled. Response to a new input or query is predicted after learning from the labeled data. All regression and classification algorithms come under the umbrella of supervised learning e.g. Logistic Regression, Decision Trees (DT), Support Vector Machine (SVM), Nearest Neighbors (NN), Naive Bayes (NB), Random Forest and Artificial Neural Network (ANN) etc. [11]. On the other hand, unsupervised learning used unlabeled data. It covers all clustering algorithms e.g. k-means clustering and hierarchical clustering etc. A supervised classification algorithm learns from the recorded labeled data and on counterpart unsupervised finds similarities between the data and separates data into groups accordingly, assign a new label to each group. Several approaches leveraged a combination of both and known as semi-supervised learning. Semi-supervised learning is a class of supervised learning tasks and techniques that also utilize unlabeled data for training. It uses a small amount of labeled data with a significant amount of unlabeled data to build the predictive model.[12]

2 Literature Review

This section describes the current trends in RE and ML. It will define a few problems that have been solved or automated using ML. This literature includes only related papers to the problem statement, which I have proposed in this document. The detailed verion can be seen in the accepted survey paper that is attached in the appendix.

2.1 Requirement classification

As per our survey [13] findings, the most highlighted problem in RE is requirement analysis. It deals with the classification of functional and non-functional requirements. Requirements documents are usually written in natural language and contain hundreds of requirements. It is hard and

time-consuming to classify them manually. These classifications are not limited to only FR and NFR, but also to the subcategories of NFR and quality attributes. This problem area can be categorized on the basis of existing datasets in the literature. In our categorization, we identify the following three classes of datasets:

- App Stores
- Social Media
- Internal Application or Software

App Stores: Millions of users share their reviews on app stores after downloading and using apps. They not only rate the apps but also write about the liked and disliked features. It is not easy to find the requirements out of such a complex data. New features identification, classification of FR and NFR, and summaries of the reviews for the improvements of an app is often done with ML [14] [15] [16] [17].

For the automated classification of FR and NFR, [14] used a total of 932,338 online reviews of the 40 top paid and free apps on appstores from top 10 different categories. Semi-supervised algorithm self- training, RASCO, Rel-RASCO for self-labeling is being used. This semi supervision technique overcomes the manual annotation problem and shows that only small amount of labeled data can achieve high accuracy. Naïve Bayes classification performs better than kNN, C4.5, and SMO.

Another study on the classification of FR and NFR was performed in [15] on 6,696 raw user reviews from iBook and 4400 raw user reviews from WhatsApp. It uses concept of augmentation of user reviews and effectively improves user reviews classification results by adding textual semantics to the sentences. The user reviews are augmented by several similar words for better classification of results. The bagging algorithm outperformed Naïve Bayes and J.48.

A classification method is produced in [16] for identifying bug reports and feature requests from user reviews. Total 146,057 reviews for 40 apps were collected from Appstore and Google Play Store. For further experimentation, 4,400 reviews are selected. The proposed model shows the upwards of 70 percent precision and 80 percent recall could be obtained using multiple binary classifier, as an alternative to a single multiclass classifier. For the classification, binary Naive Bayes algorithm is used. The results show that the commonly used NLP techniques such as stop

word removal and lemmatization could negatively affect the performance of this classification task.

In the paper [17], the primary goal is to transform online reviews into evolutionary requirements. Karplersky internet security 2011 from Amazon and mobile app of Tune-In Radio Pro V3.6 from the appstore are taken as dataset. The characteristic analysis of the reviews is considered for the automated task analysis and software relation-based propagation approach (SRPA) technique is used for the identification of opinion about common software features. Each set manually labeled the potential software feature, opinion and the polarities in the reviews, and then classify the reviews on the basis of relevant opinion semantics. For clustering, the opinion expression network algorithm Grivan Newman (GN) is used in the proposed methods S-GN. The GN algorithm produces optimized number of clusters. For the second problem, the system helped the developer and proposed a set of related evolutionary requirements for the system. This problem in this study makes it more interesting by suggesting the relevant and essential user requirements to a developer by adding polarization factor. That is not used commonly in this domain.

Social Media: Social media e.g. Twitter and Facebook have become popular platforms to gather the requirements from user posts. Users are sharing their new features requests, feedback and bug report on social platforms. Previous studies [18] [19] [20] in software engineering have applied different techniques to analyzed tweets contents related to software development. These tweets were posted by developers describing the relevant and irrelevant programming languages, libraries system methodologies tweets for software development. Our study is focused on getting user feedback for software and mobile applications for obtaining user requirements. ML has been used to classify tweets into meaningful categories like new requests and bugs etc. [21] [22] . These classifications and the information from these tweets helped the industry to know the user feedback for software improvements.

ALERTme [21] approach proposes for classifying, grouping and ranking tweets in software evolution process. For this, a total of 68,108 tweets i.e. collection of two months of tweets about Spotify, Dropbox, and Slack software dataset are used. The output is binary classification i.e. improvement request or other. For the automated classification, supervised

learning algorithm Multinomial Naïve Bayes is used. Further improvement requests are considered for the grouping which helped to sort the request and summarize them accordingly. The major contribution helped to reduce human effort to analyze each tweet for eliciting the requirements and knowing the issues of software. As the last step, these summaries and tweets are ranked by high worthy tweets. A drawback of this study is the high number of manual annotations for labeling the data as request or others. Besides, the majority voting scheme is being used to solve the disagreements. Three annotators did this process, and it took around 13.5 hours for each annotator to complete the task.

The next study [22] classify and summarize the tweets. A total of 4,000 tweets are randomly selected for ten different software. The proposed model classifies them into bug, requirement, and spam using Naïve Bayes and SVM. The results showed that 50% of data contained useful technical feedback and achieved an average classification F1 of 72% using SVM that is better than state of the art in the literature [21]. The reason is the feedback dedicated to technical stakeholders i.e. developer related tweets are focused and analyzed. Tweets dataset is labeled manually. Different techniques with vector space model (VSM)and NB are used. VSM is used for preprocessing e.g. stop word removing, sentimental analysis, stemming, and Bag of words. However, the results show that these parameters do not improve to help the results of ML algorithm for classification of the tweets. Unlike the political tweets which are polarized and carry emotions, software tweets are neutral in nature.

Internal Application or Software: The third category is dealing with requirement classification using the software product data i.e. written in Software Requirements Specification (SRS). It could be dataset for some interanl used software system. This data is composed of different software functional and nonfunctional requirements. The nonfunctional requirements have subcategories that include availability, fault tolerance, legal, look and feel, maintainability, operational, performance, portability, scalability, security and usability etc. This dataset is provided by the Requirement Engineering (RE) conference and named as Quality attributes (NFR) dataset. The size of the total data set has 625 requirements with 225 FR and rest with NFRs subcategories.

For the automated classification of FR and NFR and identification of the subcategories of NFR [23], support vector machine (SVM) algorithm is used. The data is not equally distributed because of the small number of requirements in NFRs subcategories, which are ignored. Data undersampling problem is solved by using external data i.e. user comment dataset from the Amazon, and a hybrid approach is proposed with the new dataset. This dataset contains performance and usability requirements. For the identification of specific NFRs, proposed methodology achieve the highest precision and recall for security and performance NFRs with 92% precision and 90% recall.

Two main goals are targeted in [24], first is classifying the FRs and NFRs, and second is an identification of NFR category. The data is preprocessed as a first step. Feature co-occurrence and regular expression are used to increase the weight of influential words used in NFR. The supervised learning algorithm J.48 DT is used for classifying FRs and NFRs. For achieving the categorization or classification of NFR, topic modeling using unsupervised algorithm LDA and BTM is applied. For the topic generation, the results show that BNB works better than clustering, k-means, LDA, BTM.

Another study for solving the same problem with NFR dataset with additional security dataset is conducted [25]. Software requirements are classified with the focus on security related requirements. ML algorithm conventional neural network (CNN) with a specific setting in Tensor-Flow helped to achieve the goal and better results. In all these classification problems, human input is involved for the annotation of the requirements. Semi-supervision requires less human effort in labeling requirements than fully supervised methods. The semi-supervised approach [26] using Naïve Bayes resulted in accuracy rates above 70%, considerably higher than the results obtained with supervised methods using standard collections of documents.

Most of the studies are using the app stores or Twitter data seprately for finding user requirements as shown in the table. A recent study has shown that the data from Twitter and app stores reviews can complement each other [27].This study analyzed 97.1 % cases mining tweets provided complementary information to developers about user feature request and bug report. Among all the apps 198 feature requests (22.48%) and bug reports 246(12.98%) were not reported in app store review. 40.78 % feature

requests and bug reports across tweets and app reviews were common. These results showed that complementary data sources provide better decision support. The gap in these studies are not taking advantage of other similar user discussion online platforms e.g. facebook, instagram etc, that can give more knowledge to help achieve better results. This concept is not completly explored yet, and improvements can be done by using different algorithms and techniques. Also, it is not proven which combination will work better and what algorithms will be suitable for the hzbrid data sources.

Also, the impact of these studies results in a real project is missing and not discussed clearly. It is not researched and discussed in the literature that how these studies and techniques are helping and useful in the industry. For example, if a developer is using application for getting user requirements from social media. How many user requirements were suggested by these proposed system and how many requirements were actually implemented and devolped. These results will explain is it useful or not to mine social media for gathering user requirements. In short, two primary goals or contributions can be seen to fulfill the research gap discussed above:

- First is researching the hybrid data sources e.g. tweets, appstores, linkedin etc for user requirements for building a new app or for the next version of the exisiting app.
- Second is identifying the real impact and usefullness of mining social media results in the industry.

Paper No.	Problem	Algorithm	Dataset	Mannual Annotation	Hybrid data-sources	Evaluation
[14]	classication of FR and NFR	Naive Bayes	appstore	(○)	x	x
[15]	classication of FR and NFR	Bagging	appstore	✓	x	x
[16]	classification as bug reports and feature requests	Naive Bayes	appstore	✓	x	x
[17]	Identify evolutionary requirements	Clustering with network algo	Amazon and appstore	✓	✓	x
[21]	classifying, grouping and ranking as improvement reques or other	Naive Bayes	Tweets	✓	x	x
[22]	classifying and summarize as bug, requirement, and spam	Naive Bayes	Tweets	✓	x	x
[23]	classification as FR and NFR and NFRs subcategories	SVM	PROMISE	✓	x	x
[24]	classification as FR and NFR and NFRs subcategories	J.48, Naive Bayes	PROMISE	✓	x	x
[25]	classification as FR and NFR and security related req.	CNN	PROMISE	✓	x	x
[26]	classification as FR and NFR and NFRs subcategories	Naive Bayes	PROMISE	(○)	x	x
[27]	classification as bug reports, feature request, and other	Naive Bayes, SVM.	Twitter and appstore	✓	✓	x

Table 1: Summary of state of the art

Legend: (✓) present in the study; (x) not present in the study; (○) partially present

3 Problem Statement

Conventionally, software requirement elicitation and analysis are only limited to meetings, interviews, and documented data etc. All the tasks are performed manually, requiring more effort and time. With recent data trends from various sources, the user satisfaction and opinion are more integrated into the industry. Recent studies mentioned in literature section have shown that the user analytics tools and techniques are helping developers and practitioners to deal with a large number of user feedback by filtering, classifying, and summarizing them, to decide what requirements and features they should add, change or eliminate. Different ML algorithms have been used for making it automated with different settings and parameters.Based on the literaure gap mentioned in the section above my reserach questions are:

RQ1: Can advanced classifiers or advance settings perform better than the ones used in the existing literature?
Out of various classifiers, it has been found in the literature that SVM and NB perform well for short texts. Since the user reviews about software systems are short texts, these two algorithms are usually preferred. However, it could be researched that how advanced classifiers or settings perform in comparison to SVM and NB. The comparison could be drawn on the basis of the ability of a classifier to process larger datasets, the processing time, complexity analysis and accuracy of classification. I am planning to apply neural network algorithm. However, it majorly depends on my dataset.

RQ2: What would be the impact of using semi-supervised in contrast to supervised learning?
One of the problems with the classifiers usually used SVM and NB is that they fall into the category of supervised learning. Supervised learning classifiers require a lot of human input as they work on the basis of provided training data. The training data needs to be classified by humans which is time consuming, particularly for large datasets such as the ones we are discussing. In such scenarios, semi-supervised and unsupervised learning would require less human input. Such techniques might decrease the efficiency, but for larger datasets, semi-supervised might be less time consuming than supervised learning techniques.

RQ:3 How social network sites (Twitter, Facebook and LinkedIn etc.) and online discussion forums (app stores and Amazon) reviews complement each other for developing new app or software?
The importance of the end user involvement in today's software is highlighted in the literature. However, the reusability of this large dataset is not discussed and missing. The addition of the social media data in the software development process in early stages can add global and heterogeneous perspectives. As different people from diverse cultures all around the world are sharing reviews about app or software on social media. The benefit of the large dataset is to gather information about the specific category and see what requirements users wished to have in both functional and nonfunctional perspective. A developer or practitioner without enough requirements for the project can take benefit of the existing data knowledge for specific app and domain. It will give an additional set of requirements and features and domain-specific knowledge for building the better system. I want to implement this idea to a case study to know the impact and enhancement of requirement that can be gained by this additional data.

Also, some similar features mapping can be challenging. For example, in [28] performed an initial study to observe the impact of emotional sentiment on app reviews as an informative feature and its pitfall. After researching and analyzing hybrid sources, some other informative features from one source to another can be improvized.

RQ4: How useful are the social network sites (Twitter, Facebook, LinkedIn) and online discussion forums (app stores, Amazon, google stores) reviews for requirement elicitation and what works better?
Users discuss software systems on numerous platforms including social media websites, app stores, and online discussion forums etc. However, all the discussions do not contain information specific to requirement engineering. Any information that doesn't fall into the category of features requests or bug reports generally is not of interest to us. Therefore, it is necessary to evaluate the percentage of relevant information in each of these platforms. A comparative study could be useful to evaluate which of these platforms contain more useful information as compared to rest of the platforms. Also, the hybrid models containing different sources e.g. Twitter and Amazon reviews, Twitter and app stores. What combination

works better and provide more features or bug report.

The datasets obtained from online platforms are huge. The content shared by the users in large number. However, the information would be more or less relevant to similar functionalities of the application. Going through all the extracted dataset that is classified as feature report or bug report would be humanly impossible. It is therefore imperative to summarize the obtained information. The summarization should not only group similar requests or bug reports, but also emphasize on the most discussed issues. For example, if there is a bug with a camera filter, and is discussed various times, it should be emphasized in the same manner to the developer, indicating that it requires more or immediate attention.

In this area after classification and all experiments no study has shown what was the impact of their finding and how much results were useful for devolper or practioneers. I want to perform some study with help of case study that can help to identify the impact of these finding in industry. The other missing feature in the literature is adaptability of the tool in the industry. For making this concept more practical and useful in industry there should be some tool or plugin with the entire features. The ongoing survey disclosed the fact that tools are not available or if available only taking data from one source. For this gap, plugin can be devolped. That plugin can be added to the existing RE tools used in the industry.

4 Proposed Methodology

The basic focus of the current research is to extract feature requests and bug reports for popular software systems and applications, as the discussions or user stories are readily available for them. The classification is performed on the basis of applications, and not the specific functionalities of these applications categories. However, there is not much benefit of it to the new developers or developers of less popular systems. Since the functionalities in newer or less popular systems are more or less similar to the popular applications, there could be a possibility that the information contained in the user stories of popular applications can be made to use for newer systems. The proposed methodology basically focusses on suggesting feature requests and bug reports on the basis of individual functionalities of a software system and software category. Whenever a new developer wants to create a new application, he selects the category for building his application. If devolper wants to see the new feature for

his exisiting application he will specify the application.

The database is populated based on the feedback from the classification and summarization block. The purpose of the classification and the summarization block is to take input from the existing social media and online platforms like Twitter, Facebook, App stores and LinkedIn etc. So far only single study [27] exist that takes data from two sources i.e. Twitter and app store. Otherwise, the data from these sources were dealt separately. Different steps to be performed are shown in Figure 1.

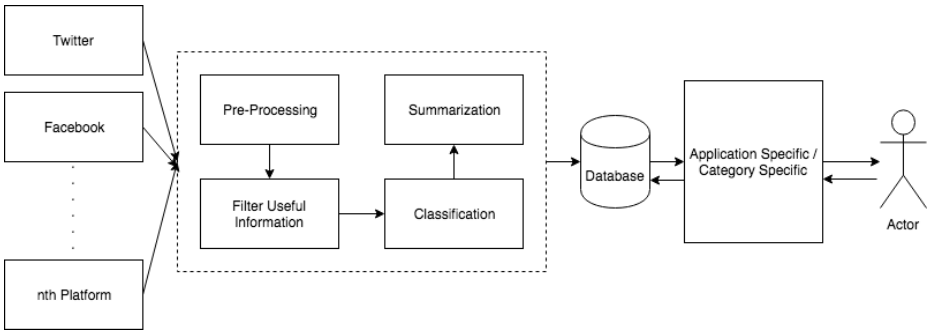

Fig. 1: Block diagram of proposed methodology.

Step 1 - Data Preprocessing: The input data will be prepared for classification after preprocessing. This step will probably include tokenization, removal of stop words, stemming, etc. This step majorly depends on the chosen dataset combination, as different social media platforms have their limitations accordingly.

Step 2 – Filter useful information: This step will gather and filter the selected App or category description data. Each application has a description on app stores and on their Twitter or Facebook page. This description will be helpful to determine the main feature of an application as relevant information. Also, some behavior feature will be learned. For example how much is the company responsive and replies to the customer.This description data will be filtered upon the user selection. It can also be used in next step as metadata.

Step 3 – Classification: Not all user discussions will be useful for feature requests or bug reports extraction. Therefore, it is necessary to filter out useful information from the other. The semi-supervised algorithm

will learn a classifier. That will classify the data into a feature request, bug reports, and other categories.

Step 4 – Summarization: Multiple discussions could be related to the same feature or bug. Since the datasets extracted from the sources is so massive, going through the complete datasets would be humanly impossible. The information needs to be summarized so that the similar type of information can be seen together. In this ranking process will consider features from the different sources such as likes, share, user behavior, sentiment analysis etc. As our dataset will be multi sourced, different features and their impact will be researched on each dataset for future selection.As the last step, all information will be clustered using topic modeling algorithm. This will give developers and practitioner an idea of which feature requests or bug reports are more emphasized and frequently discussed as compared to the rest.

The main aim of the proposed methodology is to assist developers who want to develop a new system or developers of a system with less number of users. With this technique, such developers will be able to get the advantage of the user reported features requests and bug reports from popular applications. Further, the information will be summarized on the basis of specific functionalities. So, if a developer wants to include two functionalities that exist in WhatsApp, and two functionalities that exist in Snapchat, he can easily see what are the feature requests and bug reports individually for all of these functionalities. This will hopefully result in better developed applications as required by users, quicker bug fixes even before the final release, and better testing of the applications before and after release.

4.1 User Interaction with the system:

- **Use Case 1:**
  Name: Getting requirements for building a new application
  Brief Description: Actor wants to get user requirements for building a new application in a specified category e.g. sports, entertainment etc.
  Actors: Devolpers, Requirement Analyst.
  Basic Flow:
  - Actor will select the option for building a new application.
  - Actor will choose the category in which he wants to builds a new application
  - System will collect the online data related to the specified category.

- System will apply data preprocessing on the collected data.
- System will classify data as user requirement, bug repot and other.
- System will provide a list of most influential and top ranked user requirements and bug reports in form of extractive summary.

– **Use Case 2:**

Name: Getting updated requirements for existing application
Brief Description: Actor wants to get user requirements for his existing application such as Facebook, YouTube etc.
Actors: Devolpers, Requirement Analyst.
Basic Flow:

- Actor will select the option for the existing application.
- Actor will choose his application from the list.
- System will collect the online data related to the specific application.
- System will apply data preprocessing on the collected data.
- System will classify data as user requirement, bug repot and other.
- System will provide a list of most influential and top ranked user requirements and bug reports in form of extractive summary.

– **Use Case 3:**

Name: Getting updated requirements for existing application
Brief Description: Actor wants to get user requirements for his existing application such as Facebook, YouTube etc.
Actors: Devolpers, Requirement Analyst.
Basic Flow:

- Actor will select the option comparison from the existing application
- Actor will choose his application from the list.
- Actor will choose applications for the comparison from the list.
- System will collect relevant online data for the list defined by the user.
- System will apply data preprocessing on the collected data.
- System will classify data as user requirement, bug repot and other.
- System will provide a list of most influential and top ranked user requirements and bug reports in form of extractive summary.

4.2 Evaluation:

One of the identified gaps in my research is the lack of evaluation for the existing studies. Most studies did not map their experiments result on the industry needs and impacts. That means the obtained results are similar to the manual study performed in the industry, if yes then how much? Also, how efficient results automated systems are providing to the users by considering the extra information and non-relevant information? I plan to perform this evaluation study as given below:

- The first method to evaluate the results is to draw a comparison between two different versions of one application. The specific application data will be collected before the new version. All the requirements given by the proposed system will be matched in the new version. Each identified requirement exist in the new version or not will be checked manually.

- The second method is the automated version of first method. For this two dataset before and after the new version will be compared. This method will compare which requirements are mentioned by users in the first dataset i.e. before the new version. A tracability link for these requirements will be generated in the second dataset i.e. after new version. If users has stopped asking about these requirements after the release, we assumed that those requirements are now part of the application in the new version.

- In the third method, I will perform an industrial case study that will investigate the difference in the developer requirements list result and my tool result list. The developer will mark the similar requirements in both results and give comments on comparisons. From these results, we can see the correctness and efficiency of our system. This study will help to provide the beter insight of the comparison due to the comments. That will ultimately help to provide better evaluation.

First and second casestudy has some threats to validity such as if requirement identified by my proposed system is not devolped in the new version due to some constraints e.g. orgnaizational policies, finance, scope of application etc. The third method will add the comments of devolper in the results for better anaylsis and more insight.

4.3 Timelone for thesis:

- **WP1: data collection**
  - Literature survey for finding related tools and technologies for data acquisition
  - Data collection from appstores and social media platforms e.g. Twitter, Facebook pages, and Instagram pages etc.

- **WP2: dataset creation**
  - Applying NLP techniques for pre processing
  - Identification of features
  - Labeling datasets
  - Formation of individual and hybrid datasets

- **WP3: Algorithm identification for classification**
  - Applying various ML algorithms on datasets from WP2
  - identifying suitable combinations for the hybrid dataset
  - Identifying appropriate ML algorithm for each of the dataset

- **WP4: Algorithm identification for summarization**
  - Identifying features for ranking of classified datasets
  - Applying algorithms for summary of classified datasets

- **WP5: Evaluation**
  - Perform case study1
  - Perform case study for the automated analysis for the app versions
  - Results analysis of case studies

- **WP6: Final submission**
  - Thesis Writing and submission

5 Support in Fortiss

One research project Center for code Excellence (CCE) can be considered good support in Fortiss. CCE has one part named as "Recognize and analyze trends" in which data will be collected from social media such as Twitter, and stack overflow. The collected data will be analyzed for prediction of future technology trend. I can start implementing my first task of the proposed solution and perform data analysis. For CCE

Workpackage	Description	Start Date	End Date
WP1: Data collection	1.1 Literature survey for finding related tools and technologies for data acquisition 1.2 Data collection from appstores and social media platforms e.g. Twitter, Facebook pages, and Instagram pages etc.	01/09/18	30/11/18
WP2: Dataset creation	2.1 Applying NLP techniques for pre processing 2.2 Identification of features 2.3 Labeling datasets 2.4 Formation of individual and hybrid datasets	01/12/18	30/04/19
WP3: Algorithm identification for classification	3.1 Applying various ML algorithms on datasets from WP2 3.2 identifying suitable combinations for the hybrid dataset 3.3 Identifying appropriate ML algorithm for each dataset	01/05/19	30/11/19
WP4: Algorithm identification for summarization	4.1 Identifying features for ranking of classified datasets 4.2 Applying algorithms for the summary of classified datasets	01/12/19	30/6/20
WP5: Evaluation	5.1 Perform case study in the industry 5.2 Perform case study for the automated analysis for the app versions. 5.3 Results analysis of case studies.	01/07/20	28/02/21
WP6:Thesis Writing	6.1 Thesis writing and final submission	01/01/21	31/08/21

project our implementations for data analysis can be seen as a common point. Furthermore, both side can improvize techniques according to their own needs and perspectives. Till now we both have to gather data from Twitter for analysis. However, the dataset for both of us will be different.

The other support is Dr.Levi Lucio due to his research interest in RE and ML. I am submitting a conference paper with him at the end of this week. Currently, one in-house fortiss project "User Modelling and User-Adaptive Interaction for Open Source Software Development Environment" is also under discussion. The main idea for my involvement in this project to get some hands-on experience and gaining a new research insight into my research. I will start working on the user research and analysis part. This project mainly deals with requirements on user needs, use case definition and requirements mapping and modeling user behavior and skills. This project will also help me to understand my future research integration and collaboration in AF3.

The other support is from the OCPS project. It is a requirement for OCPS project to do secondments. Secondments are two visits in other labs for research collaboration. After the completion of the above two project collaborations, I am planning to go for my first secondments.Where I will target those labs which are researching the same domain, and I can collaborate on my work with them.

References

1. Pamela Zave. Classification of research efforts in requirements engineering. *ACM Comput. Surv.*, 29(4):315–321, December 1997.
2. CHAOS Standish Group. The standish group report, 2014.

3. Gerald Kotonya and Ian Sommerville. *Requirements Engineering: Processes and Techniques*. Wiley Publishing, 1st edition, 1998.
4. Jane Coughlan and Robert D. Macredie. Effective communication in requirements elicitation: A comparison of methodologies. *Requir. Eng.*, 7(2):47–60, June 2002.
5. Didar Zowghi and Chad Coulin. *Requirements Elicitation: A Survey of Techniques, Approaches, and Tools*, pages 19–46. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
6. Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: A roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, pages 35–46, New York, NY, USA, 2000. ACM.
7. Alan M. Davis. *Software Requirements: Objects, Functions, and States*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.
8. M. Glinz. On non-functional requirements. In *15th IEEE International Requirements Engineering Conference (RE 2007)*, pages 21–26, Oct 2007.
9. Du Zhang and Jeffrey J.P. Tsai. Machine learning and software engineering. *Software Quality Journal*, 11(2):87–119, Jun 2003.
10. Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.
11. Pierre Lison. An introduction to machine learning, 2015.
12. Firdaouss Doukkali. Simple explanation of semi-supervised learning and pseudo labeling, 2017.
13. Tahira Iqbal, Parisa Elahidoost, and Levi Lucio. A bird's eye view on requirements engineering and machine learning. In *Proceedings of the 25th IEEE/ACM Asia-Pacific Software Engineering Conference (APSEC)*, APSEC 2018, 2018.
14. Roger Deocadez, Rachel Harrison, and Daniel Rodriguez. Automatically classifying requirements from app stores: A preliminary study. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pages 367–371. IEEE, 2017.
15. Mengmeng Lu and Peng Liang. Automatic classification of non-functional requirements from augmented app user reviews. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, pages 344–353. ACM, 2017.
16. Walid Maalej and Hadeer Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, pages 116–125. IEEE, 2015.
17. Wei Jiang, Haibin Ruan, Li Zhang, Philip Lew, and Jing Jiang. For user-driven software evolution: requirements elicitation derived from mining online reviews. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 584–595. Springer, 2014.
18. Leif Singer, Fernando Figueira Filho, and Margaret-Anne Storey. Software engineering at the speed of light: How developers stay current using twitter. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 211–221, New York, NY, USA, 2014. ACM.
19. P. K. Prasetyo, D. Lo, P. Achananuparp, Y. Tian, and E. P. Lim. Automatic classification of software related microblogs. In *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, pages 596–599, Sept 2012.
20. Palakorn Achananuparp, Ibrahim Nelman Lubis, Yuan Tian, David Lo, and Ee-Peng Lim. Observatory of trends in software related microblogs. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, ASE 2012, pages 334–337, New York, NY, USA, 2012. ACM.

21. Emitza Guzman, Mohamed Ibrahim, and Martin Glinz. A little bird told me: Mining tweets for requirements and software evolution. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 11–20. IEEE, 2017.
22. Grant Williams and Anas Mahmoud. Mining twitter feeds for software user requirements. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 1–10. IEEE, 2017.
23. Zijad Kurtanović and Walid Maalej. Automatically classifying functional and non-functional requirements using supervised machine learning. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 490–495. IEEE, 2017.
24. Z. S. H. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe, and K. Schneider. What works better? a study of classifying requirements. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 496–501, Sept 2017.
25. A. Dekhtyar and V. Fong. Re data challenge: Requirements identification with word2vec and tensorflow. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 484–489, Sept 2017.
26. Agustin Casamayor, Daniela Godoy, and Marcelo Campo. Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. *Inf. Softw. Technol.*, 52(4):436–445, April 2010.
27. M. Nayebi, H. Cho, H. Farrahi, and G. Ruhe. App store mining is not enough. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 152–154, May 2017.
28. Daniel Martens and Timo Johann. On the emotion of users in app reviews. In *Proceedings of the 2Nd International Workshop on Emotion Awareness in Software Engineering*, SEmotion '17, pages 8–14, Piscataway, NJ, USA, 2017. IEEE Press.