Master of Science in Software Engineering
February 2022


# Analyzing user feedback written in multiple languages and automatically identifying requirements from that feedback

**Bhargav Sakamuri**
**Rithika Vardhan**

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full-time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

**Contact Information:**
Author(s):
Bhargav Sakamuri
E-mail: bhsa20@student.bth.se

Rithika Vardhan
E-mail: riva21@student.bth.se

University advisor:
Kryztof Wnuk
Department of Software Engineering

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

# Abstract

**Background.** Requirements gathering is the most important and often an error-prone task in the software development cycle. There is no perfect procedure to get the requirements for the developing product. Every organization employs its way of extracting requirements, and the most common way is to extract requirements from user stories. Although requirements from users' stories provide an idea about what the end user expects, there is a chance that the gathered requirements are not complete. In this thesis, we suggest a way to automatically extract requirements from users' feedback tweets, a multi-lingual dataset. Requirements from feedback combined with the requirements from user stories will help the developers to understand more about what the users need and what changes they have to make to improve the product.
**Objectives.** The objective of this paper is to perform SLR (systematic literature review) to know the state-of-art techniques used to classify requirements from users' feedback in multi-lingual language automatically. Then conduct a case study on telia users' feedback which is Twitter data in Swedish and English. Build a Natural Language Processing (NLP) model that can automatically classify users' feedback.
**Methods.** The SLR (systematic literature review) is conducted using forward and backward snowballing with inclusion criteria. To build an automation model we used a case study approach to Telia's user feedback. We have used libraries like SpaCy, sklearn, pandas, re to build the model. In the developed model, we used tf-idf vectorization for feature extraction, and an SVM classifier to classify the tweets into possible requirements. The results are measured using a confusion matrix.
**Results.** In SLR, we understood the state-of-art techniques and technologies used to classify multi-lingual data. We identified that tf-idf and SVM are the perfect fit for our case study for feature extraction and classification. On conducting the case study using tf-idf feature extraction and SVM classifier, the accuracy, precision, and recall scores we obtained are 0.73, 0.67, and 0.68 respectively.
**Conclusions.** The main goal of this paper is to improve the requirements engineering phase in the software development cycle. This study will help companies who want to develop their product from users' feedback.

**Keywords:** Multi-lingual, NLP, Requirements, Automation, Tweets

# Acknowledgments

We would like to express our sincere gratitude to Prof. Krzysztof Wnuk, who served as our thesis supervisor. We express our heartfelt gratitude and appreciation to him for the guidance and assistance. His thoughtful advice and continual support have been invaluable in helping us craft the best possible document. His feedback was always constructive and gave us the confidence to take our ideas further. We would like to express our gratitude to our supervisor for teaching us how to conduct a thesis, for assisting us with the thesis topic, and for patiently guiding us through our thesis with excellent supervision and helpful criticism. The knowledge and experience he has shared with us will be of great benefit to our future academic endeavors.

We would like to take this opportunity to express our sincere appreciation to all the authors whose research articles we have used in the preparation of our thesis. Your scholarly contributions have been of immense help and have enabled us to create a comprehensive and well-researched paper. We are grateful for your efforts and dedication to your work. We sincerely thank you for your invaluable contribution to our thesis. Finally, we wanted to express our gratitude to our parents and friends for helping us both professionally and mentally by upholding our ideals and motivating us to finish our thesis.

# Contents

**Abstract** i

**Acknowledgments** iii

**1 Introduction** 1
- 1.1 Aim and objective . . . 2
- 1.2 Ethical, societal and sustainability aspects . . . 3

**2 Background** 5
- 2.1 Twitter . . . 5
- 2.2 User Feedback . . . 5
- 2.3 Machine Learning . . . 7
- 2.4 Natural Language Processing(NLP) . . . 8
  - 2.4.1 TF-IDF . . . 8
  - 2.4.2 Natural Language Tool Kit(NLTK) . . . 8
  - 2.4.3 Text Preprocessing . . . 9
  - 2.4.4 Stop words . . . 9
  - 2.4.5 SpaCy library . . . 9

**3 Related Studies** 11
- 3.1 Related work . . . 11
  - 3.1.1 Requirements Engineering . . . 11
  - 3.1.2 Multilingual Classification . . . 13
  - 3.1.3 Requirements tracking from issues . . . 13
- 3.2 Research Gap . . . 14

**4 Method** 15
- 4.1 Research Questions . . . 15
- 4.2 Research Methods . . . 15
  - 4.2.1 SLR: Snowballing Approach . . . 16
  - 4.2.2 Case Study . . . 18

**5 SLR** 19
- 5.1 Publications used in our research . . . 19
- 5.2 Research table for SLR . . . 20
- 5.3 Demographics of the selected papers . . . 25
- 5.4 Methods used . . . 25
- 5.5 How the methods were used . . . 27

5.5.1 Sentiment Analysis . . . 27
5.5.2 Support Vector Machine(SVM) . . . 32
5.5.3 Naive Bayes . . . 34
5.5.4 Latent Dirichlet Allocation . . . 36
5.5.5 Bidirectional Encoder Representations from Transformers(Bert) 37
5.5.6 Clustering Algorithms . . . 39
5.5.7 Random Forest . . . 40
5.6 Conclusion . . . 41

**6 Case Study** **43**
6.1 Data . . . 43
6.2 Data-Preprocessing . . . 43
6.2.1 Structuring the data . . . 43
6.2.2 Extracting Data . . . 44
6.3 Approach to automate classification . . . 44
6.3.1 Classifier . . . 45
6.4 Tackling Multi-Lingual data . . . 45
6.5 Implementing Classifiers . . . 45
6.5.1 Data Labeling . . . 45
6.5.2 Requirements from User Feedback . . . 46
6.5.3 Data-Labeling Credibility . . . 47
6.5.4 Case study Model . . . 48
6.5.5 Data Pre-processing . . . 48
6.5.6 Implementing Binary Classifier for each Category and Multi-class Classification . . . 49

**7 Results and Analysis** **53**
7.1 SLR Results . . . 53
7.1.1 State of the literature in analyzing user feedback for requirements engineering . . . 53
7.2 Case Study Results . . . 54
7.2.1 Case Study Data . . . 54
7.2.2 Score and Result metrics . . . 54
7.2.3 Multi-lingual Problem . . . 56
7.2.4 Classifiers Result . . . 56

**8 Discussion** **59**
8.0.1 Implementations for research and practice . . . 60
8.1 Validity threats . . . 61
8.1.1 Internal Validity . . . 61
8.1.2 External Validity . . . 61
8.1.3 Construct Validity . . . 61
8.1.4 Conclusion Validity . . . 61
8.1.5 Risk Managements . . . 63

**9 Conclusions and Future Work** **65**
9.1 Conclusion . . . . . 65
9.2 Future Work . . . . . 66

**References** **67**

**A Supplemental Information** **73**

# Chapter 1

# Introduction

Requirements are the declaration of possible choices pertaining to the functioning and quality of a system [33]. A methodical and systematic strategy to the creation and management of requirements is known as Requirements Engineering (RE). The main objectives of RE are to involve system users, identify their needs, and solicit their feedback [33].

Requirements Gathering phase is the most crucial aspect of the software lifecycle [26]. There are many ways of gathering requirements, but the current uncertainty of the technology and social environments where software systems operate has increased with web-based systems. A new way of gathering requirements involving system developers, software engineers, software users, and the general public is Data-driven Requirements [33]. In the Crowd-sourcing technique, the requirements are collected from user feedback and the Internet to collect software requirements from multiple stakeholders, clients, and consumers of the product, which is becoming a prominent method for eliciting software requirements [29]. Social media sites are useful for gathering rapid updates from individuals on relevant and existing issues. It is crucial to rapidly get user feedback, particularly following the release of new software. Whereas a new version is intended to fix bugs and include new features, unintentionally adding new bugs is also possible [37]. Manually understanding and classifying these data is time-consuming and laborious [29].

To eliminate such effort intensive work, many ways are introduced to automate the requirements gathering entirely or semi-automate the requirements classification. However, in the domain of CrowdRE(Crowd-sourced requirements), the crowd is typically many current or potential users of a software product who communicate with one another or with representatives of the software organization. Crowd-based requirements engineering aims to bring together crowd members to interact and analyze their demands as they contribute in improving contemporary software products [17]. But, Data-driven requirements and not only includes users of a software product, stakeholders but also general public [33].

Due to the large amount of feedback that organizations receive on a regular basis, classifying customer feedback is an ongoing research area. The success of software applications is frequently determined by user feedback. User feedback regarding software products can be found online, which is a potential resource for user requirements. Apps with great reviews have more significant results, visibility, downloads, and purchase rates. The widespread use of online user feedback concerning RE software applications is one potential Data-driven Requirements Engineering strategy. It is vital to identify the system requirements and to adequately document them for

a software project to be effective.

A general overview of user activity and user experience can be obtained via automatic feedback classification and also via the number of bug reports or feature requests [29]. Additionally, this can be used to compare updates over time in regards to the features offered and requested or the problem reports. This type of feedback is popular and commonly accessible to the public via online review sites and social media, and it serves as a key source of information for future software enhancements as well as modified or new requirements [33]. User feedback can be classified and filtered using analytics tools based on the information present in it. Text classification, natural language processing, and other parameters such as rating systems, text length, and text tense have all been used to classify reviews by researchers [33]. Automatically acquired usage data, logs, and interaction traces may enhance feedback quality and assist developers in comprehending and responding to feedback. However, usage data and interaction history benefit developers and analysts in effectively understanding the contexts in which users provide feedback. This automatically gathered information about software usage is referred to as implicit feedback. This information contributes to the documentation of the app, including its requirements and functionalities [33].

The main objective of our thesis is to explore how we could use this user feedback to discover and monitor requirements using machine learning. The goal is to investigate what machine learning algorithms can be used to analyze user feedback written in more than one language. The study is performed on the feedback given by the users to the Telia application. Aside from classifying the data, the challenge is to build a model that can understand and analyze the multi-lingual data with Swedish and English in it and classify accordingly.

## 1.1 Aim and objective

**Aim:**

In this research work, we aim to explore methods suitable for analyzing multilanguage data as input for requirement engineering. We concentrated on this issue since practitioners are searching for automated solutions to filter noisy feedback, detect and solve defects, and identify feature requests provided by users as motivation for future releases.

**Objective:**

- Conduct a snowballing literature review to get a better grasp of the various data mining approaches and machine learning algorithms that can be used for automated analysis of user feedback and identification of requirements from that feedback.
- Identify a method that can be used to automate the analysis of multilingual feedback in the case study and classify requirements.
- Build a model that can classify and comprehend both Swedish and English feedback.

## 1.2 Ethical, societal and sustainability aspects

The data that we have collected is from public feedback through Twitter. We will ensure to keep the data of the public confidential and will not misuse the collected data. With the current technology, every company and organization is tilting towards machine learning algorithms and models to automate manual tasks to reduce human resources and efforts. Many types of research are conducted concerning what areas in the industry can be automated and the impacts of such changes. Automating manually done work will save lots of time and be cost-efficient considering the technology shift. One of the identified processes can be automated Requirements classification from data-driven data (feedback). The data-driven requirements will provide the most accurate requirements since it involves all stakeholders and the general public [49]. But the data size of data-driven data is massive, and manually going through all data will not be a cost-efficient method. The Results of our thesis will provide a way to automate the classification of requirements from the data-driven dataset that includes multilingual s like English and Swedish. Our results will improve the requirements gathering phase, help the organization build correct products, and use public sources more effectively. The Literature review that we provide will present the state-of-art of such technologies providing readers with more information about our approach.

# Chapter 2

# Background

This chapter provides the necessary background information and context for our research, including a historical overview, definition of key terms, and relevant theories and frameworks.

## 2.1 Twitter

Twitter is a social platform that helps users communicate a huge amount of information on many different subjects through brief comments [18]. The most well-known and significant microblogging platform worldwide is Twitter. More than 600 million users have already enrolled, producing more than 500 million tweets per day [46]. With a 140-character restriction per message, end users are free to send messages. A tweet can be directed to a particular end user by using the "@" indicator and the user's username [37]. Some of these comments, called tweets, contain details that are important for software organizations and could assist engineers in determining users' requirements [18]. Users on Twitter have the option of liking and retweeting responses to tweets. Retweeting is used to forward tweets to other users, whereas "liking" is intended to express appreciation for a tweet. Retweets and likes can therefore be used as measures of how popular a tweet is with other users [37]. Retweets generally begin with the character "RT." Users have the option of tweeting directly to another user or replying to an existing tweet. "@Username" is used in replies and direct tweets to indicate who the message is for. To demonstrate their interest in a tweet's content, users can also favorite it on Twitter [46]. More than 1.5 billion messages are sent on the Twitter website every day by users. These "tweets" discuss a variety of subjects, including entertainment, sports, politics, and technology. Tweets could be related to app reviews, in which consumers promote software, discuss problems, and ask for new features. Software businesses could gain a better understanding of their users' requirements by using tweets [18].

## 2.2 User Feedback

Users provide feedback about the applications they employ via social media and application distribution systems such as app stores [39]. In comments posted on platforms where a program is accessible for download, software users can review software applications. This sort of feedback has been discovered to include useful information for software evolution in previous findings [39]. User feedback is an

essential component in developing and maintaining usable software. Software applications must adapt to their environment and usage settings in order to be properly maintained and updated [39]. In order to maximize the possibility that the software remains useful and beneficial throughout its development, it is crucial to take into account user feedback after the product release [39].

## 2.3 Machine Learning

Using machine learning as a technique to assist in the generation of requirements has become widespread [8]. Automated text classifiers have been incorporated into the requirements engineering process using a variety of techniques. According to a systematic literature review by Lim et al. [30], 38 out of 63 research studies that analyzed user feedback based on manually labeled data employed machine learning to categorize user feedback [30]. This indicates that these classifiers have become widely used throughout the field. The claimed good classification performance of some of these machine learning models is one of the possible causes of their popularity [8].

**Classification**
The act of classification involves predicting the category of a set of data points. Targets, labels, and categories are all terms that can refer to classes. The goal of classification predictive modeling is to estimate a mapping function from input variables to distinct output variables [2]. Using some training data, a classifier can determine how certain input variables relate to a certain class. Classification falls under the supervised learning category, in which the targets are also given access to the input data. Classification has numerous benefits across a wide range of industries, including target marketing, medical diagnosis, etc [2]. There are numerous classification algorithms present currently, but it is difficult to determine which one is finest [2].

**Support vector machines**
Support Vector Machines(SVM) are supervised learning models with corresponding machine learning algorithms that assess data used for regression analysis and classification [15]. Many users significantly like the support vector machine since it generates correct results with a considerable degree of efficiency. It is applied extensively to classification tasks. A dividing hyperplane serves as the precise definition of the discriminative classifier termed as a Support Vector Machine (SVM). An SVM training method creates a model that categorizes new samples according to one of two categories, given a series of training sets that have each been tagged as belonging to one of the categories. This transforms the algorithm into a non-probabilistic binary linear classifier [15].

**Naives Bayes**
A probabilistic classifier called Naive Bayes, which operates under the basic presumption that the attributes are conditionally independent, is influenced by the Bayes theorem [2]. Naive Bayes is a relatively easy algorithm to use, and in the majority of circumstances, it produces decent results. Since it operates in linear time rather than through a costly iterative approximation strategy like several other types of classifiers, it is flexible and scalable to bigger datasets [2].

## 2.4 Natural Language Processing(NLP)

The subfield of artificial intelligence known as natural language processing (NLP) enables machines to comprehend human language. The same must first undergo pre-processing because the machine cannot use it instantly. Nowadays, one of the most investigated domains is NLP, which has seen a number of ground-breaking advancements [25]. Developers from all over the globe have produced a wide variety of tools to handle natural language, and NLP depends on sophisticated computational abilities. There are many libraries available, but only a select number are widely used and hugely beneficial for carrying out numerous NLP operations. The following collection of libraries, stop words list, and code can be used to remove English stop words [25].

### 2.4.1 TF-IDF

Natural language presents a challenge because the data is in the form of raw text, which necessitates its transformation into a vector [43]. "Text vectorization" is the term used to describe the process of turning text into a vector. Because no machine learning method can comprehend a text, this is an important step in natural language processing. Text can be converted into vectors with the support of the text vectorization method TF-IDF vectorizer, which is a well-liked method for conventional machine learning methods. Text is converted into a vector form using the text vectorizer term frequency-inverse document frequency [43]. Term Frequency (TF) and Document Frequency(DF) are two ideas that are integrated. The term "frequency" refers to the number of times a particular phrase appears in a document. The frequency of a term in a document reveals its significance. Each text from the data is depicted by term frequency as a matrix, where the rows correspond to the number of documents and the columns to the number of unique terms used in each document [43]. The weight of a word is determined by its inverse document frequency (IDF), which attempts to decrease the weight of a word if its occurrences are distributed over all of the documents [43].

### 2.4.2 Natural Language Tool Kit(NLTK)

The Natural Language Toolkit is a collection of software modules, data sets, and tutorials that facilitate research in machine learning and NLP [4]. Python is used to develop NLTK, which is released under the GPL open-source license. The efficiency with which items of interest can be tallied in particular circumstances is made possible by NLTK's capabilities for conditional frequency distributions. Research on text categorization may benefit from such data [4]. NLTK offers interactive graphical user interfaces that let users monitor program state and track program execution in detail. The majority of NLTK components offer a demonstration mode that allows them to carry out an intriguing task without any additional user input. Even small changes to programs can be performed in response to hypothetical situations [4].

### 2.4.3 Text Preprocessing

It is the process of getting text data ready for machines to use for operations like analysis and prediction, among other things. There are many other processes involved in text pre-processing [25]

### 2.4.4 Stop words

Stop words are the words that are typically eliminated from natural language processing. The text does not get much information from these terms, which are among the most common in any language . Stop words in English include "the," "a," "what," "so," "that", "my", "just" etc [25]. Any human language has an excess of stop words. By getting rid of these words, we can make our text more focused on the key information by eliminating the low-level information. The elimination of such phrases has no detrimental effects on the model we train for our task. Stop words should be eliminated since they increase the dataset size, which increases training time due to the larger number of tokens involved [25].

### 2.4.5 SpaCy library

SpaCy is a Python library for extensive natural language processing [50]. SpaCy is an open-source, free library that is ideal for individuals who deal with a significant amount of text. It allows you to develop apps that need to handle a large amount of text and is intended for usage in development [50]. SpaCy can be used to design a system for text pre-processing before deep learning, natural language interpretation, and information retrieval. SpaCy provides a variety of functions and features, ranging from machine learning features to language notions [50]. The functions that SpaCy provide are tokenization, dependency parsing, parts of speech tagging (POS), Entity Linking(EL), text classification, training, and many more [50].

# Chapter 3

# Related Studies

This chapter provides an overview of relevant studies and research in the field of requirements Engineering and machine learning, serving as a foundation for the analysis presented in subsequent chapters. This chapter also consists of research gap which explains about identifying and examining the existing gaps in the current body of knowledge surrounding analysis of multilingual feedback, setting the stage for the research objectives and questions to be addressed in the following chapters.

## 3.1 Related work

### 3.1.1 Requirements Engineering

Requirements Engineering is the most critical aspect of software engineering. According to the authors' paper [29] from past studies, they found that there is approximately 60 percent of all errors occur due to inefficient requirement engineering. Crowdsourcing is a well-suited way of gathering requirements in this continuously changing world. In the paper [29], the author wants to automate the classification of crowdsourced requirements and suggests a way to classify spam and requirements using project-specific and non-project-specific keywords as features. The authors selected KeePass, Mumble, Winmerge, open-source projects from sourceforge.net. They [29] prepared 1000 user requests for each project and labeled the requirements and most of the requirements related to capability and usability. Then compared the results with implemented machine learning algorithms in weka using five equal fold cross-validation. The machine learning models used are K-nearest neighbors, SVM, Naive Bayes binary, and multiclass for classification.

Classifying crowdsourced requirements is time-consuming and laborious work. The amount of data gathered with crowdsourcing is enormous and filled with spam, and manually reading and classifying is also a waste of time. But crowdsourcing requirements will give precise requirements as it involves all stakeholders and end-users, known as the crowd. So, the paper [51] proposed a model to automate the classification requirements from Crowdsourced data. The model involves classifying feedback (crowdsources) using data mining and classifying the data into functional and non-functional requirements. The tool used for classification and data mining is rapid miner. Using 60% of data for training (labeled) and 40% for testing, the author ran the model using the TF-IDF technique [51]. The models used by the model are Naive Bayes and decision trees. From the experiment conducted by the author,

the result came out with 54% functional requirement and 46% non-functional requirements with 92% and 84% of accuracies with Naive Bayes and decision trees, respectively [51].

Several APP distribution platforms, such as the Apple App Store and Google Play, offer a review feature that enables users to review and comment on an APP after using it, allowing users to send feedback to the APP's developers and vendors. App distribution refers to the process of making an app available to a large number of people in order to increase app activation and utilization [58]. App marketers frequently seek for app distribution platforms as a means of promoting their app, through paid advertising or any other way. With the rise in the number of user reviews, determining how to effectively and efficiently assess them and discover prospective and significant user demands in order to develop APPs has become a problem. Hui Yang et al [58] proposed a method for automatically identifying requirements information from user reviews and further classifying them into functional and non-functional requirements, based on a combination of information retrieval technique (TF-IDF) and natural language processing technique(regular expression), with human intervention in keyword selection for requirements identification and classification [58]. Hui Yang et al. evaluated the proposed approach using user reviews from a popular APP called iBooks in English App Store and looked into the cost and return of the approach, such as how the size of the sample reviews for keyword selection (cost) affects classification results in precision, recall, and F-measure (return) [58].

Binkhonain et al [3] presents the results of a systematic review of 24 machine learning (ML)- based techniques for detecting and classifying Non - functional requirements. The absence of labelled datasets was indicated as the first major barrier in the review [3]. ML-based techniques have worked effectively, with more than 70this research. The processes used by the reported ML-based approaches to discover and classify NFRs in requirements documents are shown in this study [3].

Santos et al [45] presented the initial findings of a systematic mapping study of the classification strategies employed in crowd-requirements engineering experiments that aimed to classify user feedback for RE using NLP techniques. This study provides a brief overview of the current landscape of automated classification techniques for requirements engineering, with the hope that the findings will motivate researchers to try new approaches towards further research in this area. Santos et al. conducted a systematic literature review to gain a thorough and extensive analysis of the literature on identifying user feedback while avoiding any selection bias and gaps in their research [45]. This research was purely descriptive, with only the ML algorithms and core ML features being compared. Even though this research focused on the study of user feedback, the performance of machine learning models in different contexts and not just within RE could demonstrate which strategies are most suitable [45].

### 3.1.2 Multilingual Classification

Researchers employed supervised machine learning and contrasted deep learning and traditional machine learning techniques. Despite acquiring many labels, their findings show that traditional machine learning can still produce results comparable to deep learning. Christoph Stanik et al [49] applied supervised machine learning fed with crowd-sourced annotations to analyze 10,000 English and 15,000 Italian tweets from telecommunications Twitter support accounts, as well as 6,000 English app reviews. Christoph Stanik et al. proposed a set of classification experiments for identifying requirements-related information in English app reviews and also English and Italian tweets [49].

### 3.1.3 Requirements tracking from issues

The paper "Analyzing and Relating Bug Report Data for Feature Tracking" by Michael Fischer, Martin Pinzger, and Harald Gall explains how requirements can be extracted from bug reports. The authors use natural language processing and text mining techniques to analyze bug reports and identify patterns and relationships between bugs and features. By doing so, they demonstrate how bug reports can be used to extract requirements for software systems [13]. The paper shows that bug reports contain valuable information about the software's features and requirements, and that this information can be used to improve the software's quality and reliability. The authors conclude that bug reports can play an important role in the software development process by providing valuable feedback on the functionality and usability of the software, and by helping developers to better understand the needs and expectations of users [13]. Overall, the paper provides valuable insights into the relationship between bug reports and requirements, and highlights the importance of using bug reports as a source of information for requirements extraction [13].

Walid Maalej et al. introduced several probabilistic algorithms to classify app reviews into four categories: bug reports, feature requests, user experiences, and text ratings [32]. To classify, they combined review metadata like star ratings and tense with text categorization, natural language processing, and sentiment analysis tools. They ran a lot of experiments to examine the accuracy of techniques and to see how they compared to basic string matching. Walid Maalej et al [32] discovered that relying just on metadata leads in low classification accuracy. When paired with simple text classification and natural language text preprocessing — particularly with bigrams and lemmatization—classification precision for all review categories increased to 88–92% and recall increased to 90–99%. Single multiclass classifiers were outperformed by multiple binary classifiers [32].

## 3.2 Research Gap

The concept of data-driven requirements has the potential to help in the requirements elicitation process [22] [33]. Data-driven requirement identifies requirements from multiple public resources like public feedback, user, and other stakeholder feedback [33].

The given data for requirements is a large and diverse, making manual classification and analysis economically unfeasible. Several authors have suggested ways to automate the requirements classification process using machine learning [49], Deep learning [49] [29], data mining techniques [51], etc.

Even though we found many techniques to automate the classification process of the data-driven requirements, we found very few papers that consider the involvement of multilingual data in the dataset [49]. There are no papers and methods for classifying multilingual data involving Swedish and English. The collected data for the data-driven requirements will be from the general public and crowd (users and stakeholders). It is not given that the public should provide their feedback in English only. The data collected by the native public and crowd will be a combination of both Swedish and English. Since there are no models that classify the requirements from multilingual data with both Swedish and English, we consider finding a model that best suits to classify requirements from multilingual data(Swedish and English) as a potential research gap to continue our research.

# Chapter 4

# Method

This chapter details the research questions and methodology employed in our research, including the selection of the methods, data collection methods, and justification for choosing the methods.

## 4.1 Research Questions

**RQ1:** What is the state of the literature in analyzing multilingual user feedback for requirements engineering?

**Justification:** The reason for selecting the RQ1 is to find state-of-the-art technologies to classify requirements from users' feedback, and to explore the methods, tools that researchers use to categorize tweets that are both monolingual and multilingual.

**RQ2:** What method can be used to automate the analysis of multilingual feedback?

**Justification:** Since the feedback data from Telia is in both Swedish and English, The RQ2 is selected to find a method from the SLR study to build a model that is the best fit for the case study data.

## 4.2 Research Methods

The methodology we selected to find the state of literature in analyzing user feedback for requirements engineering is a Systematic Literature Review(SLR). A systematic review of the scientific literature in a particular field is critical for identifying research problems and justifying future studies in that field [52]. The term "snowballing" describes the practice of identifying other papers using a paper's reference list or citations. Snowballing, on the other hand, could benefit from a systematic means of looking at where articles are actually referenced and cited, in addition to looking at reference lists and citations [57]. Reference lists are simple to analyze, and when combined with the place and context of the reference, it is usually simple to discover relevant papers [57].

During snowball searches, we discovered keywords and generated search strings. The first challenge in using a snowballing strategy is determining the starting set of papers to employ for this procedure [57]. By using Google Scholar, we identified a

good starting set of papers. We looked at the first 200 results on Google Scholar using the search string "classifying user feedback in requirements engineering" and selected the papers that are relevant to our current study, which is included in the related work section. Using well-known scientific databases like IEEE, Springer, ACM, Science Direct, etc. on software engineering research, we used our search string to find relevant papers on classifying feedback for requirements engineering. We attempted to identify additional new papers to include in our research as a part of the snowballing strategy using the reference lists of the selected papers. Data extraction is carried out on all identified papers in accordance with the research questions stated in the systematic literature review. We could achieve a better understanding of the various data mining methodologies and machine learning algorithms that can be used to analyze user feedback data and identify requirements from that feedback through a literature review. One of the key benefits of snowballing is that it begins with relevant papers and then uses them to guide additional research [57]. The previous work that is accessible serves as the foundation for knowing what strategies have been employed in analyzing user feedback for requirements engineering in this research. These research results and conclusions may aid in achieving a number of methods and algorithms that can be used to analyze feedback and identify requirements from that feedback as a motivation for future releases.

We conducted a systematic literature review (SLR) as part of a wider benchmarking study to get a thorough overview of the literature on classifying user feedback. We discovered that our collection of systematically gathered literature works through snowballing, which presents and employs a variety of divergent classification methods. The papers were chosen based on the three inclusion criteria (IC) given below. A paper meeting (IC1 or IC3) and IC2 were included in the selection, whereas a paper meeting of one of any three was excluded from the selection.

**IC1:** About multilingual machine learning
**IC2:** About machine learning for analyzing user feedback (any feedback from social media or general) But EMPIRICAL so real data
**IC3:** Mining Twitter (because it is a short text maximum of 160 characters so the feedback is short)

Snowballing has resulted in a total of 75 papers. This number included papers for which we were unsure if they satisfied our selection criteria. The number of articles for data extraction was decreased to 29 after more reading of the abstract, introduction, and conclusion parts, to which IC2 and (IC1 or IC3) inclusion criteria for snowballing were applied.

### 4.2.1 SLR: Snowballing Approach

The steps of the snowballing process are listed and explained in the following subsections.

**START SET:**
Identifying keywords and generating search strings is the initial stage of database

searches. Finding a starting set of papers to employ for the snowballing technique is the first task when using a snowballing approach. A preliminary start set is produced by any search for articles to include in the starter set. The only papers in the preliminary start set that will ultimately be included in the systematic literature study constitute the real start set [57]. We've chosen a start string "classifying user feedback in requirements engineering" associated with our topic for our investigation to get an efficient and precise start set. In databases like Google Scholar and the Bibliotek Library, we used this search string to look for publications with a related topic. We have first gotten a start set of 11 papers for snowballing. After discussing with our supervisor, we decided to limit our start set to three papers.

**ITERATIONS:**
We began the first iteration of backward and forward snowballing after choosing the start set, which will only comprise articles that will be considered in the analysis [57]. Before opting to select a paper as one of the final papers in the snowballing process, the entire paper must first be analyzed. The inclusion criteria proposed helped us to select the papers on that basis [57]. Our supervisor guided us throughout this process. After implementing forward and backward snowballing on the start set, we got a total of three iterations.

**BACKWARD SNOWBALLING:**
Backward snowballing is the practice of identifying relevant papers to include in our research from the reference list [57]. The first thing we did was go through the reference list and eliminate any articles that didn't meet our inclusion criteria. The papers that have already undergone analysis due to being identified earlier through either backward or forward snowballing in this iteration or a previous one are then excluded from the list [57]. The other papers are relevant for inclusion after these have been eliminated. This process resulted in 16 papers that satisfied our inclusion criteria from the reference list of Start Set papers.

**FORWARD SNOWBALLING:**
Identifying new papers based on the papers that cite the one being studied is known as "forward snowballing" [57]. Google Scholar is used to look for the citations in the paper that is considered for our study. The abstract is looked at first, and if it isn't enough, the area citing the work that is already included is inspected [57] . If this is not adequate, the complete text is examined to determine whether to accept the new paper. The method used to review the papers is comparable to how papers were selected by employing backward snowballing [57]. This process helped us in identifying 10 papers through forward snowballing.

New papers found in the iteration are gathered for the following iteration after backward and forward snowballing. To achieve transparency, it is critical to work through iterations one at a time [57]. This process resulted in a total of 29 papers, which satisfied our proposed inclusion criteria for our snowballing study.

**WHY INCLUSION CRITERIA?**
During our snowball study in the start set, we found 3 papers. It is an extensive

start set, and having a significant start set will extend the required time to conduct our study, and it will also lead to diverging the research from the goal. To have a confined and goal start-set, we have introduced the three inclusion criteria- IC1, IC2, IC3 and included the papers that satisfy IC2 and either one from IC1 or IC3. With the inclusion criteria, we narrowed down the start-set to 3 papers. Then in Iteration one, we found 61 papers in total, and with (IC2 and (IC1 or IC3) ), it got narrowed down to 13. Similarly, 11 out of 20 from iteration two and two from iteration 3.

### 4.2.2 Case Study

The methodology we selected to find the methods to automate the analysis of multilingual feedback to generate requirements is a case study approach.
**justification:**
Case Study, An intensive analysis of an individual unit (as a person or community) stresses developmental factors concerning the environment [14]. A case study is a well-recognized research methodology that focuses on a single complex problem [48]. Case studies are, by definition, conducted in real-world scenarios with a high degree of realism [44]. Unlike many other research methodologies, a case study process is flexible with design and parameters [44]. The data used for the case study research is usually qualitative, not quantitative [48] [44]. Quantitative data is numerical data that describes the statistical significance, whereas qualitative data involves words, descriptions, pictures, and diagrams [44]. The case study research method prefers qualitative data over quantitative as qualitative data provides a more detailed and deeper explanation making it easier to analyze the case [44]. We collected qualitative feedback on the telia application through Twitter for our thesis. It is an empirical case study with the data directly available from Twitter.

The Case Study methodology aims to conduct Exploratory research and Improving [44]. This case study will explore the state-of-art of existing models and methods close to classifying multilingual data into requirements. Comparing those methods found in Literature review with our data set, we chose to create a new model that takes the collected feedback on telia as input and classifies the requirements. As part of Exploratory research, we found a few papers that used crowdsourced data to generate requirements. The papers mentioned in Related works chapter used different methods, models, and tools to classify crowdsourced data. Still, we did not find any paper that used Swedish as a multilingual dataset. We then continued our search to find models that suited our requirements and found a few libraries with Swedish vocabulary (XLM-R, spaCy) [7] [1]. Throughout this thesis, we explore more research papers using the snowball technique and try out models to achieve the aim of the study.

# Chapter 5

# SLR

This chapter presents a systematic literature review of the current state of research on the methods and tools employed by the researchers to analyse user feedback, providing an in-depth analysis of relevant studies, theories, and findings in the field.

## 5.1 Publications used in our research

The distribution of the publications seen in the papers is shown below in Figure 5.1. The papers chosen for our study are from well-known journals, which are listed below. The well-known journals are, namely, Springer, arXiv, ACM, and IEEE. There are about 18 papers selected for our study from the IEEE, 4 from the ACM, 5 from Springer, and 2 from the arXiv publication.

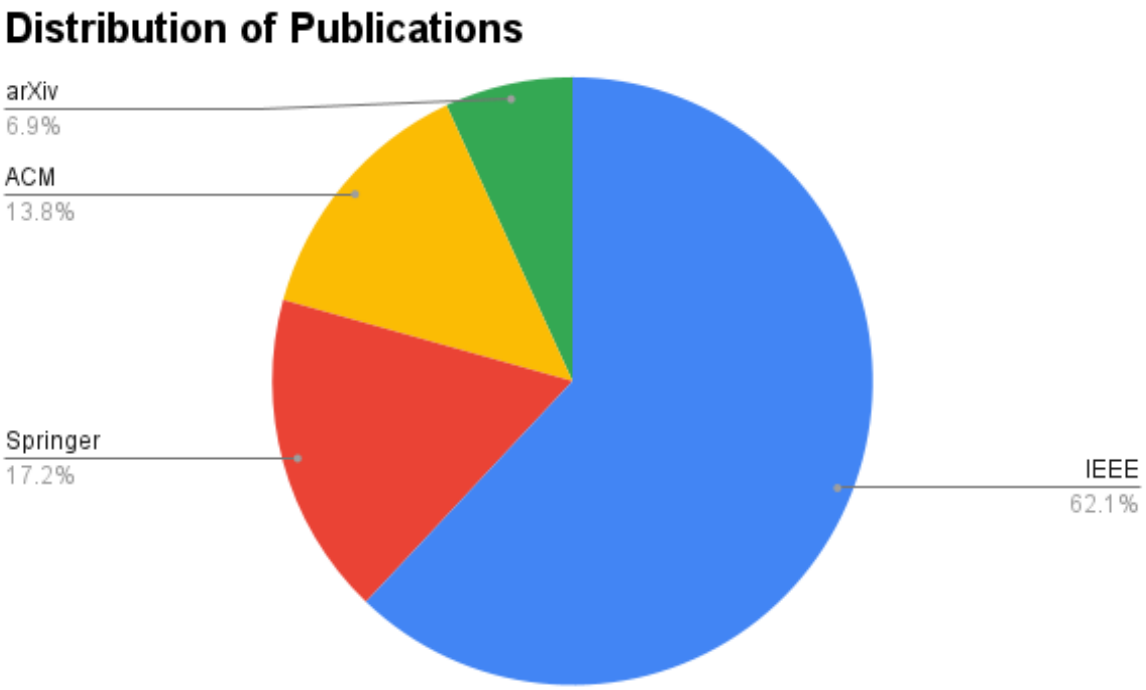


Figure 5.1: Distribution of Publications

## 5.2 Research table for SLR

The research table shows all 29 articles we chose for our research study as a part of the literature review. The table also has a column that provides a short description of all the methods and techniques employed by the authors in their respective articles.

The number of articles we chose for the start set, first iteration, second iteration, and third iteration based on the specified inclusion criteria is shown in Figure 5.2.

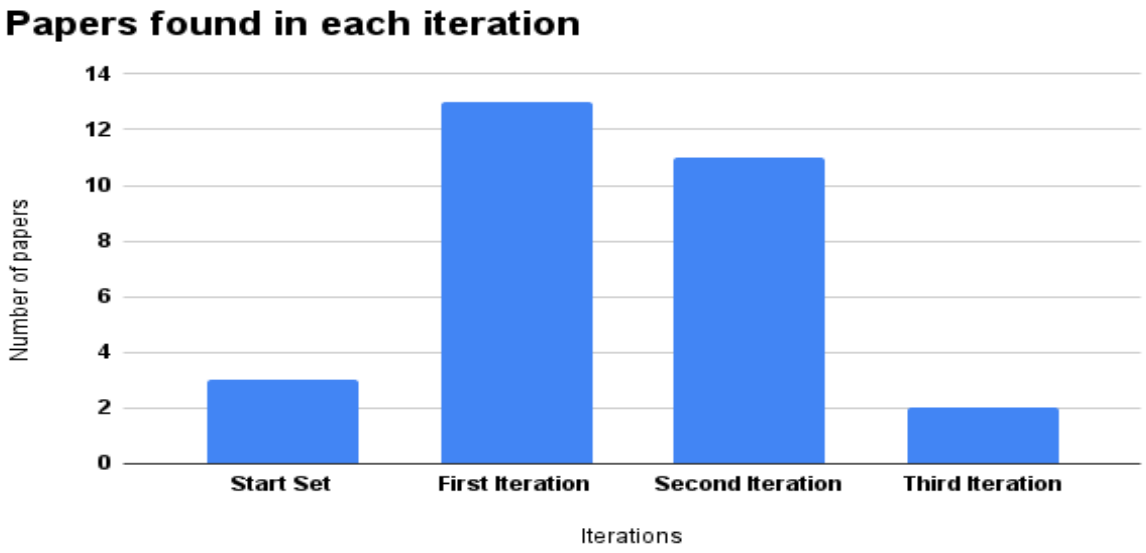


Figure 5.2: Papers found in each iteration

**IT1** - First Iteration

**IT2** - Second Iteration

**IT3** - Third Iteration

**SS** - Start Set

**Sn** - Papers found in the start set

**rn** - Papers found through references

**cn** - Papers found through citations

| S NO | ID | Iteration | Title | short description | Year of Publication |
|---|---|---|---|---|---|
| 1 | S1 | SS | Classifying Multilingual User Feedback using Traditional Machine Learning and Deep Learning | By considering keywords, sentiment of the tweet, tense of the statement and TF-IDF as features the author have developed a convolutional neural network to classify the feedback [49] | 2019 |
| 2 | S10 | SS | A Little Bird Told Me: Mining Tweets for Requirements and Software Evolution | ALERTme approach (preprocessing, classification, grouping, ranking), topic modelling and Biterm Topic Model (BTM) for grouping semantically related tweets, Multinomial Naive Bayes (MNB) for classification [19] | 2017 |
| 3 | S8 | SS | Mining Twitter Feeds for Software User Requirements | Bag of words (BOW), stemming (ST) and stop-words removal, Sentiment Analysis, SVM and Naives Bayes using Weka to analyze and classify the feedback [56] | 2017 |
| 4 | r21 | IT1 | How Do Users Like This Feature? A Fine-Grained Sentiment Analysis of App Reviews | PART 1: Collocation finding algorithm, Lexical sentiment analysis, topic modeling algorithm-Latent Dirichlet Allocation; PART 2: created a truth set through a careful, manual, peer-conducted content analysis process [20] | 2014 |
| 5 | r30 | IT1 | Analysis of User Comments: An Approach for Software Requirements Evolution | Information Extraction techniques, topic modeling, Aspect and Sentiment Unification Model (ASUM)- an extension of Latent Dirichelt Allocation LDA, K median clustering classification. It includes manual classification of topics for two apps and for facebook app , the topics were generated using ASUM [5]. | 2013 |

| | | | | | |
|---|---|---|---|---|---|
| 6 | r14 | IT1 | Automatic classification of nonfunctional requirements from augmented appuser reviews | Identifying and having to consider non-functional requirements (NFRs) present in user reviews. They merged four classification techniques Bag of words, TF-IDF, Chi-Squared with three machine learning algorithms Naive Bayes, J48, and Bagging to classify user reviews [31]. | 2017 |
| 7 | r33 | IT1 | NIRMAL: Automatic identification of software relevant tweets leveraging language model | NIRMAL (For preprocessing and ranking the tweets) and keyword-based approach. They manually labeled 300 tweets as either software related or not and used a part of the labeled tweets as training data to learn a classifier using SVM. They downloaded these tweets using the Twitter REST API [46]. | 2015 |
| 8 | C12 | IT1 | Same Same but Different: Finding Similar User Feedback AcrossMultiple Platforms and Languages | SIMBA approach, which finds similarities between the feedbacks. They used machine translation, sentiment analysis, Google translate API, Naives bayes, tf-idf, manually labelled tweets and text classification, to extract the sentiment polarity and content nature of user feedback written in different languages [39]. | 2020 |
| 9 | r35 | IT1 | Short text classification in twitter to improve information filtering | Their proposed approach classifies incoming tweets into categories based on the author's information and features within the tweets. Manually labeled tweets with matching categories, Bag Of Words (BOW). They used Naïve Bayes classifier in WEKA using 5-fold cross validation [47]. | 2010 |
| 10 | r31 | IT1 | A needle in a haystack: What do twitter users say about software? | They used an open-source library to access the Twitter Search API to import tweets written in English. Content analysis techniques, sentiments of the tweets were identified [18]. | 2016 |
| 11 | c16 | IT1 | Transfer Learning for Mining Feature Requests and Bug Reports from Tweets and App Store Reviews | Transfer Learning, usage of monolingual and multilingual BERT models (Comparison for user comment classification in italian and english tweets), Binary Relevance to solve multi class classification [21]. | 2021 |

| | | | | | |
|---|---|---|---|---|---|
| 12 | r40 | IT1 | Automatic Classification of Software Related Microblogs | Framework to identify microblogs (3 components- Webpage crawler, Text processor and machine learning tweet classifier), stop word removal using NLTK, Porter Stemmer algorithm for stemming of tweets, SVM to classify unlabelled tweets (tweets related to a task of engineering software systems) [42] | 2012 |
| 13 | c17 | IT1 | App Review Analysis via Active Learning | Active Learning (ML paradigm)-classification Framework to classify reviews, Bag of Words (BOW) approach, stop words removal, lemmatizing the tokens, naive Bayes, SVM, logistic regression, Scikit-learn implementation but only Naive Bayes classifier results were reported in this paper. Baseline (BL) and Active learning classification variants were trained [10]. | 2018 |
| 14 | r39 | IT1 | How Can I Improve My App? Classifying User Reviews for Software Maintenance and Evolution | NLP, Text analysis, Sentiment Analysis, manually analyzed user reviews into a set of categories, stop words removal, stemming (English snowball stemmer), tf(Term frequency), Naive Bayes for predicting the sentiment, AR-Miner for creation of the truth set [41]. | 2015 |
| 15 | r34 | IT1 | Spam detection on Twitter using traditional classifiers | Random forest, Naive Bayesian, KNN, SVM to classify spam data [35] | 2011 |
| 16 | c19 | IT1 | Using frame semantics for classifying and summarizing application store reviews | Researchers analyzed how efficiently semantic frames classified useful user reviews into different categories of software maintenance requests. SVM using Weka and Naive Bayes was used to classify the data. BOW, BOF, IteratedLovins Stemmer for stemming [24]. | 2018 |
| 17 | r17 | IT2 | Release planning of mobile apps based on user reviews | CLAP (Crowd Listener for release Planning), CLAP uses the Weka implementation of the Random Forest machine learning algorithm to classify user reviews, stop words removal, Porter stemmer for stemming, N-grams extraction, Reviews are clustered using DBSCAN clustering algorithm [53]. | 2017 |
| 18 | c1 | IT2 | Preprocessing Rolein Analyzing TweetsTowards Requirement Engineering | 11 preprocessing techniques on the classification performance of requirements-related tweets, NLTK stemmer, WordNet Lemmatizer, Multinomial Naive Bayes (MNB), Decision Tree (DT) and Random Forest (RF) as the classification algorithms, scikit-learn package to implement the classifiers [12]. | 2019 |

| | | | | | |
|---|---|---|---|---|---|
| 19 | c2 | IT2 | Extracting and analyzing context information in user-support conversations on twitter | Auto-Populate Issue Trackers with Structured Bug Reports Including Context Items mined from User Tweets, crawled tweets using Twitter search API, pre-processing the tweets of the crawled dataset by removing conversations including non-English tweets using the LangID library, Creation of dataset using tool doccano (Text annotation tool for labelling), summative analysis, NLP using spaCy, fastText library, an algorithm to extract app and system versions from the text [34]. | 2019 |
| 20 | c3 | IT2 | Mining User Opinions in Mobile App Reviews: A Keyword-based Approach | MARK approach, K mean clustering algorithm, tf idf weighing scheme, developed a custom filtering algorithm for non-English reviews, word stemming, POS tagging, Stanford lemmatizer, stemming the algorithm, Keyword expanding algorithm [54]. | 2016 |
| 21 | r5 | IT2 | App store mining is not enough for app improvement | Sentiment analysis, Topic modeling, Lemmatization, Sentiment analysis, Naive Bayes, SVM(Automatic classification), used an open-source module, named Pattern, which is built on top of the NLTK comprehensive Python package, Twitter API. Latent Dirichlet Allocation (LDA) to extract topics from app reviews and app related tweets, crowdsourcing [38]. | 2018 |
| 22 | c4 | IT2 | Mining Reddit as a New Source for Software Requirements | Qualitative manual analysis performed on Reddit dataset, SVM, Random Forest, Naive Bayes, NLTK tool kit for preprocessing, stemming, BOW, Tf-idf, SVM outperformed other classifiers in this study [23]. | 2021 |
| 23 | r6 | IT2 | Recommending and Localizing Change Requests for Mobile Apps based on User Reviews | CHANGE ADVISOR approach(that analyzes the structure, semantics, and sentiments of sentences contained in user reviews to extract useful user feedback), AR Doc a review classifier(combines Natural Language Processing (NLP), Sentiment Analysis (SA) and Text Analysis (TA) techniques), Stop words removal, Stemming(Porter Stemmer), tf-idf, BOW, NLTK, TextBlob python libraries for text analysis, PYENCHANT library for spell check, POS tagging classification, three clustering algorithms- LDA, application of Genetic Algorithms to LDA (LDA-GA), Hierarchical Dirichlet Process (HDP) algorithm [40]. | 2017 |
| 24 | c5 | IT2 | Evaluating Unsupervised Text Embeddings on Software User Feedback | Topic modelling, averaged word embeddings, word frequency embeddings, and pretrained text embedding model embeddings, Latent Dirichlet Allocation (LDA), Biterm model, Gibbs sampling algorithm for a Dirichlet Mixture Model (GSDMM), Tf-idf, BOW, Stop words removal, NLTK package, pre-trained embedding models (such as LaBSE and S-BERT) [9]. | 2021 |
| 25 | c6 | IT2 | Evaluating Software User Feedback Classifiers on Unseen Apps, Datasets, and Metadata | Huggingface's Tokenizer library for tokenization, "bert-base-uncased" version of the "Bert Tokenizer [8]. | 2021 |
| 26 | r8 | IT2 | Analyzing and automatically labeling the types of user issues that are raised in mobile app reviews | Researchers described a method for automatically assigning multiple labels to user reviews. They manually labeled and analyzed user reviews, Stopwords removal, Porter stemming algorithm, MEKA- multi-labeling classification tool, Binary Relevance, Classifier Chains, SVM, Naive Bayes, J48 for classification [36]. | 2015 |
| 27 | r10 | IT2 | What Would Users Change in My App? Summarizing App Reviews for Recommending Software Changes | Combined Intent Classification and topic extraction [11]. | 2016 |
| 28 | r1 | IT3 | Are tweets useful in the bug-fixing process? An empirical study on Firefox and Chrome | Normalization process, stop words removal, Porter stemmer, Standford Parser, POS tagging, N-grams extraction using NLTK tool kit, Latent Dirichlet Allocation (LDA) [37]. | 2017 |
| 29 | r2 | IT3 | Why People Hate Your App — Making Sense of User Feedback in a Mobile App Store | Researchers presented an approach WisCom can automatically summarize and understand user reviews. Latent Dirichlet Allocation to analyse user reviews, Sentiment Analysis, linear regression model [16]. | 2013 |

## 5.3 Demographics of the selected papers

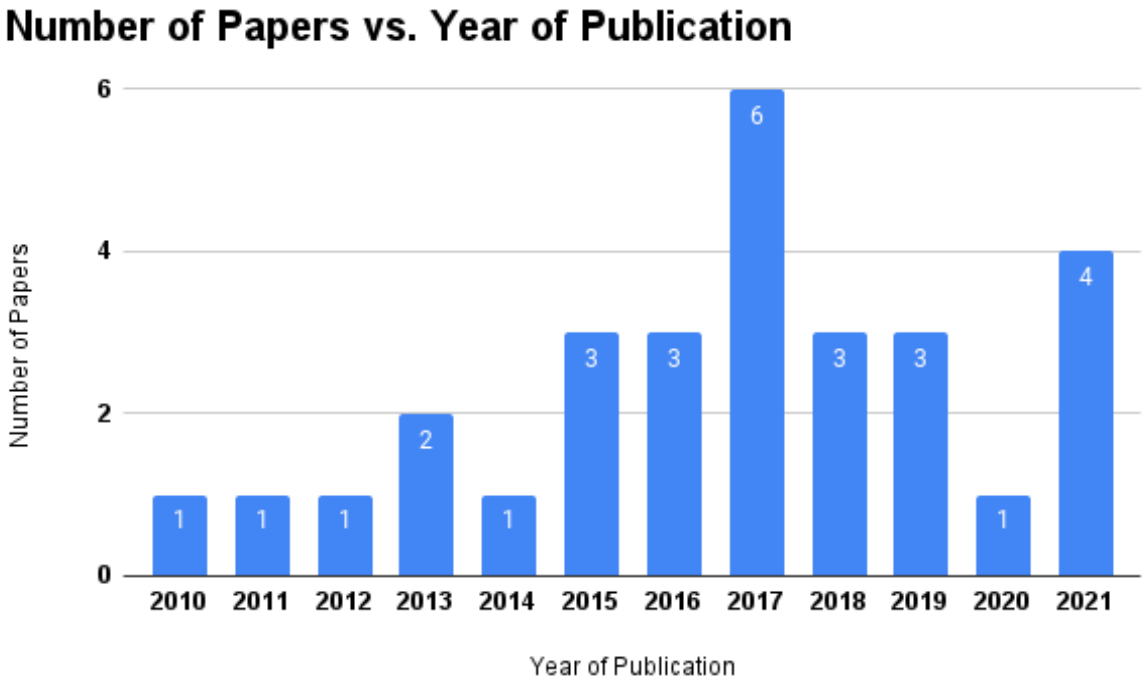


Figure 5.3: Number of Papers and their year of publication

We found 29 papers that focused on how to automatically extract requirements from users' feedback. While most papers discuss automatically extracting data from a single lingual language, i.e. English, only two articles discuss multilingual language requirements extraction. The authors have used different approaches to automate the requirements extraction from feedback, and the following sections in this chapter will discuss the research papers. The labels on bar graph in Figure 5.3 represent the number of papers published in the specified year on the X-axis.

## 5.4 Methods used

- **Machine learning techniques:** Machine learning is the first word that comes to any technical mind when someone talks about automation. Machine learning is a collection of algorithms that can predict or automate tasks. The machine learning algorithms used by the authors in automating the requirements gathering from users' feedback are SVM, Naive Bayes, Multinominal Naive Bayes, K median clustering classification [5], Sentiment Analysis [56] [20] [38] [11] [39] [41] [40] [18] [49], J48 [36] [41], Bagging to classify [31], Topic modeling [20] [38], multi-label classification, C4.5 algorithm [18], DBSCAN clustering algorithm [53], logistic regression [41], Random Forests [35], and Decision

trees [18].

- **Deep learning techniques:** Deep learning is a subset of machine learning. It is a process to teach computers to do what comes naturally to humans, i.e. learn by example. Since the requirements gatherers manually identify requirements, it seems logical to replace humans with machines to reduce the time and effort in gathering requirements. The methods we referred to in the papers are Artificial neural networks, monolingual and multilingual BERT models [9] [21] [8], Convolutional neural networks [49], Active learning [10], and transfer learning [21].

- **Tools and process models:** Tools and processes are some pre-existing models that help deal with the problem in a structured manner. The tools and processes used in the papers are ALERTme [19], Biterm Topic Model [19], NIRMAL [46], SIMBA [39], Doccano text, AR-Miner, SURF [11], Weka [35] [53] [47] [24] [56], MARK [54], CHANGEADVISOR [40], WisCom [16], Doccano and CLAP approach [53].

- **Natural Language Processing(NLP):** Natural language processing refers to the ability of the computer to process text and analyze human language [25]. NLP will be the core of this paper. Since the purpose of the research is to understand the text and deduce requirements, understanding the text is essential. The papers that we referred to have used NLP in different contexts using different techniques and models, which are the Sentiment of the tweets, i.e. sentiment analysis, Lexical sentiment analysis, Latent Dirichlet Allocation [16] [37], Aspect and Sentiment Unification Model, stop words removal [56] [18] [42] [10], lemmatizing the tokens [20] [38] [49] [31], stemming, TF-IDF [18] [41] [39], Porter stemmer for stemming [40] [53], N-grams extraction [49] [46] [53], NLTK stemmer, WordNet Lemmatizer [20], Snowball stemmer [11] [41] [36] [54] LangID library [34], fastText [49] and spaCy [49].

The bar graph below i.e. Figure 5.4 displays the frequency with which the mentioned techniques were applied in the articles we chose for our SLR.

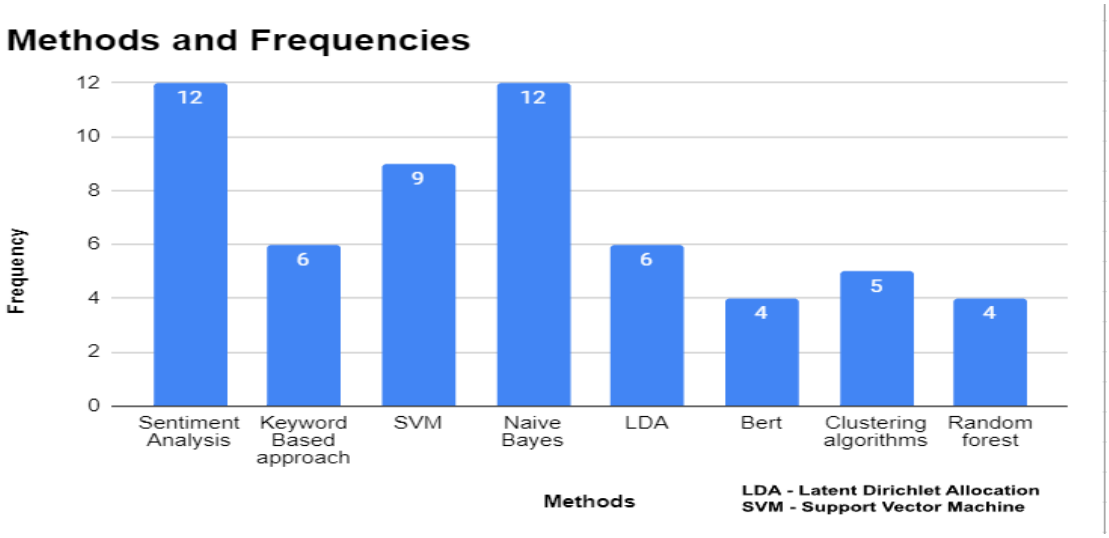


Figure 5.4: Methods and their frequencies

## 5.5 How the methods were used

In this section, we present how researchers conducted automation on user feedback to gather requirements. The data is directly from users as feedback, containing lots of noise and redundancy. There should be a preprocessing phase to prepare the data depending on the automation technology used to perform automation on such data. After preprocessing the data, an appropriate technique, either with machine learning, deep learning, NLP, or tools and process models, is used to reach the goal.

### 5.5.1 Sentiment Analysis

Researchers have used different algorithms in machine learning, as mentioned in section 5.4. Grant Williams et al. and Anas Mahmoud et al have collected 4000 tweets from 10 software and used a machine learning text classifier for categorizing technically informative tweets [56]. The authors have manually identified 4000 tweets using a Twitter hashing search engine an API and used two machine learning algorithms Support vector machine(SVM) and Naive Bayes(NB) to answer the research question "To what extent can informative software-relevant tweets be automatically classified?". In the NB approach, Williams et al. and Anas Mahmoud et al defined the features of the model as the individual words of the text, which is known as a Bag Of Words (BOW) where the data is represented in a 2-dimensional word x document matrix. While a Bernoulli NB model has a matrix that contains a binary 0 or 1 depending on the existence of a specific word, a multinomial NB, on the other hand, uses normalized frequencies of the words in the text for word x document matrix [56]. SVM algorithm is used to find optimal hyperplanes to separate unique or similar patterns into classes or a category [56]. The data is classified into an N-dimensional space and then placing the data on either side of the hyperplane to classify the tweet. Williams et al. and Anas Mahmoud et al. have included features like Bag of Words(BOW) a simple un-ordered collection of words, and some

text pre-processing techniques like Stemming (ST) and Stop word (SW) removal and Sentimental Analysis to improve the accuracy of the classifiers. To Implement SVM and NB, Williams et al. and Anas Mahmoud et al. have used the Weka tool. Since the goal of the paper is to know to what extent can informative software-relevant tweets be automatically classified the data is pre-classified by the researchers manually and then evaluated the classifiers using f1 score, recall, and precision [56]. Concluded that the textual content of Twitter messages is the only element that influences classification accuracy [56]. Other attributes that are frequently utilized as supplemental attributes to improve Twitter data are irrelevant and SVM and NB are the robust classifiers for Twitter data [56].

Researchers proposed an automated method for developers to filter, gather, and analyze user evaluations in this paper [20]. They discovered fine-grained app features in the reviews by using natural language processing techniques. The user sentiments about the detected features were then extracted and given a general score among all the reviews. Furthermore, they employed topic modeling techniques to aggregate fine-grained features into higher-level features that were more relevant. They started by gathering user reviews for a particular app and extracting the title and text comments from each one [20]. The text data is then preprocessed to eliminate noise before feature extraction. The features are then extracted from the reviews using a collocation finding algorithm and gathering features according to their significance. Each sentence in the review is given a sentiment score by lexical sentiment analysis [20]. To construct a more coarse-grained summary, they used Latent Dirichlet Allocation (LDA), a topic modeling algorithm on the extracted features and their associated sentiment ratings. They employed the Natural Language Toolkit's (NLTK4) part of speech (POS) tagging functionality to identify and extract the nouns, verbs, and adjectives in the reviews, stop words removal, Lemmatization using the WordNet lemmatizer from NLTK for preprocessing. SentiStrength was used for Sentiment Analysis [20]. They also constructed a truth set by conducting a manual, peer-reviewed content analysis. These insights can assist developers in analyzing and quantifying users' opinions on certain app features, which they can then use to identify new requirements or plan future releases [20].

In this paper [38], We look at how Twitter may help with mobile app development by providing more insights. Henry Cho et al. found strong relationships between the number of reviews and tweets for most apps after evaluating 30,793 apps over the course of six weeks. Furthermore, researchers were able to extract 22.4 % more feature requests and 12.89 % more bug reports from Twitter using machine learning classifiers, topic modeling, and subsequent crowd-sourcing. The CliPS Pattern module provides the Twitter API, which was used to gather tweets about the apps [38]. Duplicate tweets such as retweets and similar tweets were eliminated. The second phase involves using machine learning techniques to classify tweets, followed by computationally complex topic modeling. The SVM (Support Vector Machine) and Naive Bayes classifiers were used to filter out reviews and tweets that were classified imprecisely. Pattern, an open-source module developed on top of the NLTK (Natural Language tool kit) comprehensive Python package, was used. This program collects tweets using the Twitter API and analyzes their sentiment [38]. Because the contexts

of the review and tweet are so significant, they selected lemmatization over Porter's stemming. To extract topics from app reviews and app-related tweets, they employed Latent Dirichlet Allocation (LDA). The findings of this study revealed that social media mining could help app developers learn more about their users' needs [38].

Researchers developed SURF (Summarizer of User Reviews Feedback), a revolutionary method for summarizing the massive amount of data that developers of popular platforms must maintain on a regular basis as a result of user feedback. For collecting user needs, SURF uses a conceptual model that is valuable for developers undertaking maintenance and evolution activities [11]. They evaluated SURF on user reviews of 17 mobile apps and 5 of which were built by Sony Mobile, conducted with a total of 23 developers and researchers. Researchers believe that integrating topic extraction, intention classification, and the ability to incorporate well-defined maintenance actions for particular areas of an app will assist developers plan future software updates and fulfill market requirements [11]. To automatically mine user intentions, SURF uses an approach through a Machine Learning (ML) method, so an Intent Classifier integrates Natural Language Parsing (NLP), Sentiment Analysis (SA), and Text Analysis (TA) techniques for identifying sentences in user reviews that are relevant from a maintenance standpoint [11]. They created an NLP classifier to allocate a sentence in a review to one or more categories automatically. The preprocessing phase involved stop words removal, and filtering irrelevant sentences The Snowball Stemmer Algorithm was used to stem each sentence. SURF generates summaries using a sentence selection and scoring process depending on the maintenance feedback, review topics, review length, and popular features using clusterization [11]. SURF additionally uses labels to highlight the semantic category to which each sentence belongs, for example, a bug, feature request, question, or any other piece of information, with the goal of assisting developers in the planning and execution of certain maintenance tasks [11].

In the paper [39], Emanuel Oehri et al. automatically presented an approach to group similar feedback into clusters. Emanuel Oehri et al. achieved group feedback in multiple languages. The feedback is taken from Apple App Store, Google Play App Store, Twitter, and Facebook and written in English, French, German, and Spanish [39]. To group similar feedback in multiple languages, the authors have used SIMBA (SIMilarity Based Approach) approach. In this SIMBA, all the non-English feedback is converted into English feedback, and then the sentiment polarity of each feedback using a lexical sentiment analysis tool [39]. The similarity score is calculated and compared between feedback with the help of a weight function [39]. This weighted function is a combined result of text similarity, classification, and sentiment analysis [39]. To translate non-English feedback to English, the authors used Google Translate API [39] and then implemented text pre-processing to improve their results [39]. The Multiple Naive Bayes is used for classification, and this training is done on a vector space model using TF-IDF [39].

The paper written by E. Guzman [41] introduces a taxonomy for classifying app reviews into categories relevant to software maintenance and evolution, and also a method that combines three techniques, including Natural Language Processing,

Text Analysis, and Sentiment Analysis, to automatically classify app reviews into the stated categories. Researchers manually analyzed user reviews of seven Apple Store and Google Play apps and constructed a taxonomy of relevant content for software maintenance and evolution. To train ML techniques, they also experimented with different combinations of NLP, Text analysis, and sentiment analysis features. To decrease the number of text features for the ML approaches, stop-word removal and stemming using the English Snowball Stemmer are used [41]. They employed the tf (term frequency) indexing instead of the tf-idf indexing since the inverse document frequency (idf) excludes terms that occur in a large number of reviews. The Stanford Typed Dependencies (STD) parser was used to automatically detect sentences meeting the specified NLP standards [41]. They evaluated a variety of machine learning techniques, including the conventional probabilistic naive Bayes classifier, Logistic Regression, Support Vector Machines, J48, and alternating decision trees. The findings of this research suggest that combining NLP, TA, and SA approaches enables app developers to identify significant sentences with reasonable precision and recall [41].

In the paper [18], the authors have considered 10,986,494 tweets about 30 different desktops and mobile software applications from three other distribution platforms, apple's AppStore and Google Play, from the ten most downloaded software applications [18]. This data is manually annotated by the researcher [18]. The authors have conducted an experiment to classify tweets according to their relevance to the identified stakeholder group. This experiment involves training machine learning models to classify the tweets based on labels referred to as multi-label classification. The method employed by the authors is a binary relevance method where the classifier for each label is trained [18]. The classifiers C4.5, a decision tree algorithm, and a Support Vector Machine (SVM) are used to classify the text [18]. To perform classification, the authors have prepared the data with some preprocessing techniques such as converting text data into tokens, removing stopwords, removing numerical characters, and "#" and "@.". Also removed URLs and replaced them with a link to know that there exists a URL. To increase the model's accuracy, the authors have used TF-IDF (Term Frequency Inverse Document Frequency) as a weighting scheme. As for training, it is done in ten-fold cross-validation [18]. Precision, recall, and F-measure is used as evaluation metrics for this experiment.

Palomba et al. [40] proposed CHANGEADVISOR, a novel approach that examines the structure, semantics, and sentiments of sentences in user reviews in order to extract relevant user input from maintenance aspects and recommend modifications to the developers. ARDOC, a review classifier, is used by CHANGE ADVISOR to automatically mine feedback from user reviews. ARDOC uses a Machine Learning (ML) algorithm to integrate Natural Language Processing (NLP), Sentiment Analysis (SA), and Text Analysis (TA) approaches to identify user feedback that falls into one of four categories: Information Giving, Information Seeking, Feature Request, and Problem Discovery [40]. A standard Information Retrieval (IR) normalization procedure is used to normalize the retrieved code components. The methods like English stop words removal, Stemming using Porter stemmer, Tokenization, Singularization, Repetitions Removal, POS tagging classification, short tokens removal, short docu-

ments removal, and term frequency-inverse document frequency (tf-idf) were used as a part of preprocessing [40]. This process generates a list of bag-of-words, for each of the app's classes. Using the Python libraries NLTK and TEXTBLOB, two sets of tools for Natural Language Processing (NLP), and the spell check PYENCHANT library, researchers built a customized IR process to convert the end user language into input appropriate for textual analysis. They experimented with three different clustering strategies for user feedback clustering: Latent Dirichlet Allocation (LDA), Genetic Algorithms to LDA (LDA-GA), and Hierarchical Dirichlet Process (HDP) [40]. The initial evaluation of CHANGEADVISOR revealed that the method offers a considerable advancement over the current state-of-the-art in app review mining, as far as the researchers are concerned [40].

Christoper Stanik et al. [49] compare the traditional machine learning approach with deep learning to find the best model for classifying user feedback (tweets and app reviews) written in English and Italian into problem reports, inquiries, and irrelevant. Researchers gathered around 5,000,000 English and 1,300,000 Italian Tweets directed to telecommunication organizations' Twitter assistance accounts. To avoid uncertainty, researchers pre-processed the data in three steps. The first step was to convert the text to lowercase, which reduced ambiguity by normalizing it. The second step adds masks to specific keywords i.e it masks account names, links, and hashtags. Lemmatization, the third phase, normalizes the words to their root form. The length (in words) of the written user feedback was extracted. Using the sentistrength library, researchers analyzed the sentiment of the user feedback. They provided complete user feedback, such as a tweet as the library's input [49]. The library then produces two integer values, one ranging from -5 to -1 to signify the negative feedback, and the other ranging from +1 to +5 to denote the positive feedback. The Italian language model allowed four tenses when obtaining the tense using spaCy, however, researchers had to determine the tense for the English language by extracting the part-of-speech tags. Researchers experimented with 30 feature combinations, including "sentiment + fastText" and "n-words + keywords + POS tags + tf-idf." They conducted a Grid search for hyper-parameter tuning [49]. When it comes to English app reviews, standard machine learning outperforms deep learning when it pertains to the f1 score. When optimizing for f1, the findings for the Italian tweets reveal that deep learning achieves greater precision while traditional approaches obtain a higher recall. The f1 score demonstrates that all approaches are similarly effective. The findings of this study suggest that traditional machine learning can produce equivalent outcomes to deep learning in their context [49].

**The summaries written about all the 9 articles mentioned above show that researchers have commonly used sentiment analysis to improve the accuracy of the classifiers in those research studies. They also used many algorithms which include topic modelling, LDA, Decision Tree, SVM and Naives Bayes. They used various NLP techniques to analyze the user feedback**

### 5.5.2 Support Vector Machine(SVM)

In the paper [42], Philips K. Prasetyo et al. conducted a study to investigate the feasibility of automatic classification of microblogs as relevant or irrelevant to engineering software systems. To classify the tweets authors have proposed a framework in which the data are tweets and the text from webpage crawler [42]. For training, the authors have manually labeled the tweets as relevant or irrelevant and used them to train the classifier [42]. The trained model is a discriminative model the authors used it for classifying unlabeled tweets at the testing phase [42]. To improve the accuracy there is a text pre-processing phase where the common stopwords for each tweet are removed using NLTK [http://nltk.org] [42]. Then the text is tokenized and limited to the word's common root form. On this pre-processed text data the authors have used SVM (Support Vector Machine). For this experiment, the authors used 300 tweets and divided them into 10 folds each with 30 tweets, and used nine of them for training and 10 for testing. The results of the experiment are evaluated using accuracy, precision, recall, and F1 measure [42].

Yuan Tian et al. [46] proposed NIRMAL, a unique technique that automatically identifies software-relevant tweets from a collection of tweets, to aid developers in dealing with noise. This method assesses the consistency of content in order to distinguish between significant and meaningless tweets [46]. An N-gram language model was created using SRILM, a prominent language modelling toolkit.SRILM takes a set of documents and a parameter N as input and produces an N-gram language model that describes the regularities of text in the input set. The collected tweets were subjected to some basic preprocessing [46]. Punctuation marks and URLs were eliminated, and all words were converted to lowercase. The accuracy@K scores for the keyword-based method were then evaluated using a random sample of 200 tweets. After that, they used NIRMAL to sort all 227,225 keyword-containing tweets and manually assessed the top-K tweets to estimate the accuracy@K score to see whether NIRMAL might enhance the accuracy of the keyword-based strategy [46]. The findings of the experiment demonstrate that NIRMAL can achieve a good accuracy score, which is 192 percent higher than the accuracy score of a random model [46].

Researchers presented a simple unsupervised method for extracting fundamental context items from tweets, such as the platform, device, app version, and system version, using predefined keyword lists, word vector representations, and text patterns [34]. This technique, when linked with a chatbot that seeks missing context information from reporting users, intends to auto-populate issue trackers with structured bug reports [34]. They crawled tweets using the Twitter Search API in the data collection phase. Researchers gathered tweets from the official Netflix, Snapchat, and Spotify help accounts for this study. Using the LangID library, they pre-processed the crawled dataset's tweets by eliminating conversations, including non-English tweets [34]. The tweet texts were then converted to lowercase, with line breaks, double whitespaces, and mentions of support account names removed. They then generated a truthset using a tool doccano with labelled tweets from Netflix, Snapchat, and Spotify support accounts [34].

This paper [36] investigates the multi-labeled nature of reviews of 20 mobile apps in the Google Play Store and Apple App Store. Researchers discovered that up to 30% of reviewers identify many categories of issues in a single review. Researchers presented a method for automatically assigning multi-labels to user reviews in this study. They downloaded 226,797 one-star and two-star user reviews from the Apple App Store and 3,480 one-star and two-star user reviews from Play Store [36]. Only user reviews with a one- or two-star rating were manually labeled. By manually inspecting app reviews, they were able to identify problem categories. They merged the title and comment for all user reviews. Apart from hyphens and apostrophes, they eliminated all numbers and special characters. Words that appear less than three times in the data set are filtered out. They employed the snowball stemmer to stem the words because it uses the Porter stemming algorithm [36]. To generate a dictionary of all words that remained after preprocessing, they utilized the String To Word Vector filter, which is provided in MEKA, a multi-label classification tool that is an extension of WEKA. They used tf-idf and experimented with various approaches which include Binary Relevance (BR), Classifier Chains (CC), and Pruned Sets for multi-labelling [36]. They chose SVM, J48, and NB classifiers based on previous research showing that these classifiers perform well with multi-labeled data for automated classification of manually labelled data. This study shows that, despite the challenges, we can efficiently and automatically label reviews to address real-world stakeholder issues [36].

Iqbal et al. [23] identified reddit as a new potential data source and investigated whether and how gathering user feedback from reddit may assist requirements engineering and software evolution. Researchers examined the ability of automatic classification and employed machine learning algorithms to classify unstructured and noisy reddit data into three categories: bug reports, feature-related data, and irrelevant data [23]. To get access to the Reddit API, they used the Python package PRAW. The authors of this study carefully evaluated the content of posts in order to learn about software requirements and development procedures. They used the Natural Language Toolkit(NLTK) to pre-process the Reddit data (NLTK). They used two methods for vector conversion: Bag of Words (BoW) and Term Frequency and Inverse Document Frequency (TF-IDF). They used the Support Vector Machine (SVM), Random Forest (RF), and Naive Bayes (NB) algorithms to identify the right classifier for the dataset [23]. The dataset was unbalanced, thus researchers utilized the oversampling approach Synthetic Minority Over-sampling Technique to fix it (SMOTE). The findings of this study show that SVM performs better than the other classifiers with an F-score of 84% in all the settings [23].

Researchers examined how well semantic frames performed when categorizing useful user feedback into different categories of reasonable software maintenance requests [24]. They also presented and analyzed the effectiveness of several summarization algorithms in producing short and precise summaries of informative reviews. The reviews were manually classified into feature requests, bug reports, and other categories by the authors and a judge. They used SVM and Naives Bayes to classify the data. They employed Weka which is a data mining software that uses a wide range of machine learning and classification methods [24]. SEMAFOR, a probabilis-

tic frame semantic parser, was employed by them to automatically process English phrases and produce semantic annotations in a unique XML format. They employed the IteratedLovinsStemmer provided by Weka to stem the reviews for the BOW evaluation [24]. In terms of classification performance, SVM using the BOF (Bag of Frames) representation was able to outperform naive Bayes. They employed MARC 2.0, a software that facilitates a data collecting function that allows users to download recent reviews from the Apple App Store. The findings of this paper demonstrate that semantic frames can support a precise and effective review classification procedure [24].

**Researchers have employed Support Vector Machine(SVM) in 4 articles mentioned above. Yuan Tian et al. [46] employed both SVM and keyword-based approaches to identify software-related microblogs. In a few articles [24] [23] [36], SVM has outperformed other classifiers by giving good results. The five articles also used several algorithms, but SVM resulted in good performance compared to the other methods.**

### 5.5.3 Naive Bayes

Bharath Sriram, David Fuhry, and Murat Demirbas et al. [47] proposed a method that determines class labels and a set of features on Twitter, with an emphasis on user objectives such as daily chatter, chats, sharing information/URLs, and so on. When contrasted to the TweetStand, this approach is broader. Based on the author's information and properties inside the tweets, it aims to classify incoming tweets into categories such as News (N), Events (E), Opinions (O), Deals (D), and Private Messages (PM). They obtained a sample of recent tweets from random individuals and filtered out those that were not in English. Their final collection includes 5407 tweets from 684 different authors [47]. Experiments are carried out using 5-fold cross-validation with the existing implementation of the Naive Bayes classifier in WEKA. They identified seven features (8F), including one nominal (author) and seven binary features. Experiments demonstrate that even without meta-information, classification accuracy is high, and the proposed approach exceeds the standard Bag-Of-Words(BOW) strategy i.e. the results reveal that the BOW strategy performs reasonably well, but the 8F approach outperforms it immensely [47] .

Emitza Guzman et al. [19] demonstrated ALERTme, a system for classifying, grouping, and ranking tweets concerning software apps. Machine learning techniques have been used to automatically identify tweets demanding changes. Topic modeling is used to cluster semantically similar tweets, and a weighted function is used to rank tweets based on various factors, such as content category, sentiment, and the number of retweets. Tokenizing all tweet content, converting all text to lowercase, extracting n-grams with a one to a three-word length where these n-grams are only utilized in the classification stage, eliminating stop-words, and stemming the text are all steps involved in the data preparation process [19]. For the classification task, researchers employed Multinomial Naive Bayes (MNB) as it outperformed other machine learning algorithms for classifying text. For categorizing semantically similar tweets, they

employed the Biterm Topic Model (BTM), a topic modeling algorithm specializing in a short text. They employed SentiStrength, a lexical sentiment analysis tool that concentrates on short informal text, to determine the mood indicated in a tweet [19]. They compared the Multinomial Naive Bayes classifier's results to those of a Random Forest classifier, which performed well when classifying user comments from app reviews. The classifiers were trained and tested with the help of Weka. The results of the BTM method were compared to those of the Latent Dirichlet Allocation (LDA) technique. On average, BTM outperformed LDA. This study demonstrates that the ALERTme technique can automatically classify tweets into improvement requests and other categories [19]

In this paper [12], researchers proposed an approach that entails over 6000 experiments to systematically examine the effects of eleven preprocessing approaches on the classification performance of requirements-related tweets. They chose three prominent classification algorithms and conducted 6144 experiments on a dataset of 4000 software tweets to see how eleven different PTs(Preprocessing Techniques) affected classifier performance. These three algorithms are trained to classify tweets into three different categories: "bug report," "feature request," and others [12]. This study examined a data set of 4000 labeled tweets from ten popular software products. Among classification algorithms, they chose Multinomial Naive Bayes (MNB), Decision Tree (DT), and Random Forest (RF). They employed the scikit-learn package to generate the classifiers, which includes a collection of well-defined interfaces for implementing machine learning algorithms. Ebrahimi et al. [12] explored the impact of conducting individual PTs on the performance of three classification algorithms. When it comes to the best individual PT for each classifier, rm-5(Removing numerics) for the DT algorithm and rm-2 for both the MNB and RF algorithms outperformed the others [12].

In the paper [31], Mengmeng Lu et al. stress the importance of non-functional requirements and how the developers should learn all non-functional requirements to deliver a complete product. Mengmeng Lu et al. found that user feedback will be an excellent source for gathering all non-functional requirements [31], and going through all feedback will be laborious and time-consuming. Mengmeng Lu et al. has proposed a model that automatically classifies user feedback into four types of non-functional requirements: reliability, usability, portability, and performance; Functional requirements, and others [31]. To categorize the feedback, Mengmeng Lu et al. combined four classification techniques, BoW, TF-IDF, CHI square, and AUR-BoW (Augmented User Reviews BoW), with three machine learning algorithms Naive Bayes, J48, and Bagging, to classify user reviews [31]. The data is taken from App Store [31]. By eliminating stop words and preprocessing the data by lemmatization, stemming, and sentence split, the text is sent to the classification phase and training and then to testing, which is done tenfold [31].

To reduce the laborious work involved in App review analysis, Venkatesh T. Dhinakaran et al. proposed to use active learning, a machine learning paradigm framework like a pipeline that minimizes human effort by selecting unlabelled reviews for labeling via uncertainty sampling [10]. The process to analyze and categorize the feedback is done in three steps. An active learner that iteratively chooses unlabelled

reviews for labeling and group similar reviews as a cluster [10]. Second, A human reviewer reviews the clusters and labels the [10]. The last phase is where there is a review classifier that learns from the set of tagged reviews and automatically predicts the category, and generates the labels for new reviews [10]. Venkatesh T. Dhinakaran et al. [10] developed four binary classifiers regarding feature request, bug, user experience, rating, and multiclass classifier with reviews from all four classes [10]. The training is done incrementally, starting with 20% data from the start of the dataset for training and 30% data from the end of the dataset for testing. The reviews are pre-processed by removing stop words and lemmatizing the tokens; this is a Bag of Words approach, and each token is considered a feature. Venkatesh T. Dhinakaran et al. mentioned that they did not use feature selection which is one of the steps in the pipeline to measure the effectiveness of active learning [10]. Venkatesh T. Dhinakaran et al. selected three machine learning algorithms- naive Bayes, logistic regression, and SVM (Support vector machine) for classification. The results are measured using precision, recall, and F1 scores.

**The articles described above demonstrate how well the Naive Bayes algorithm classified user feedback. In an article [19], the authors have used Naives Bayes to classify the tweets. Their selection is based on Naive Bayes classifiers' outstanding text classification performance compared to other machine learning algorithms. In this article [10], researchers have employed naive Bayes, SVM (Support Vector Machine), and logistic regression classification algorithms but only the results of naive Bayes are reported. The articles [47] [31]have used Naive Bayes machine learning algorithms combined with other classification techniques, namely BoW, TF-IDF, CHI2, and AUR-BoW for the classification process.**

### 5.5.4 Latent Dirichlet Allocation

Mezouar et al. [37] proposed a methodology for filtering out noisy tweets and mapping the remaining tweets to bug reports. They performed an empirical study to see how beneficial Twitter is in the bug-fixing process. They started by downloading bug reports from issue-tracking systems such as Bugzilla, and then they crawled Twitter for tweets. By cleaning and normalizing the obtained data, they preprocessed both the bug reports and the tweets [37]. They built an automated tool that collects, preprocesses, and selects meaningful Twitter feedback. Secondly, they used a text-similarity off-the-shelf search engine to map the bug reports and tweets [37]. The bug reports preprocessing phase includes Non-English words removal using Moby words list, Stop words removal, and Stemming using Porter stemmer. The tweet preprocessing phase includes correcting anomalies, filtering out irrelevant tweets and Normalizing tweets. They employed the Stanford parser's Part-Of-Speech tagging to find instances of negated bug-related terms [37]. Using the Natural Language Toolkit(NLTK), they extracted the complete set of n-grams from the entire collection of bug-reporting tweets [37]. They extracted topics using the Latent Dirichlet Algorithm(LDA). Researchers discovered that at least 33% of Firefox and Chrome bugs can be detected early using timely tweets from end-users following a new release [37].

Researchers proposed WisCom in this paper [16], a system that can evaluate massive amounts of user ratings and reviews in mobile app stores at comment-word-centric analysis, app-centric analysis, and market-centric analysis levels. Researchers used a Breadth-First-Search method to crawl all of the web pages holding app description information, beginning with Google Play's home page [16]. Then, using an open-source Google Play API, they crawled all of the user reviews for each app. To perform comment-level sentiment analysis, they developed a regression model based on the terminology people used in their reviews [16]. The preprocessing phase includes the removal of HTML tags, the removal of comments with more than 5 non-ASCII characters to filter out non-English reviews, the Conversion of letters to lowercase, the removal of uncommon words, Using space and the appropriate delimiters, breaking the comment strings into words. The relationship between review text and rating was modeled using a linear regression model [16]. Researchers analyzed user reviews using the Latent Dirichlet Allocation model to discover important root causes. They added three more steps to the preprocessing phase, which includes filtering the noisy data. They solely chose negative comments associated with 1-star or 2-star ratings to identify common reasons why users are dissatisfied with certain apps. Next, in the linear regression model, they filtered out words that earned a non-negative weight to better focus on the users' negative sentiments [16]. The Stanford Topic Modeling Toolbox was used to train an LDA model. WisCom was able to identify inconsistencies in user comments and ratings, pinpoint the main reasons why users dislike an app, and track how users' complaints evolved over time [16].

**The above summaries show that the authors have used Latent Dirichlet Allocation(LDA) to extract the topics to analyze user reviews.**

### 5.5.5 Bidirectional Encoder Representations from Transformers(Bert)

The classification of bug reports and feature requests is one of two frequent classification tasks being evaluated in this research [8]. We study the effectiveness of classifiers when assessed on feedback from different apps than those present in the training set and when evaluated on totally distinct datasets using 7 datasets from previous research. There are 707 feedback comments in the smallest dataset and 4,385 in the largest [8]. The datasets contain different types of customer feedback in the form of tweets, forum postings, and app store reviews. All feedback text was initially tokenized before being used to train and test deep pre-trained language model-based classifiers. This was accomplished by utilizing the "bert-base-uncased" variant of the "BertTokenizer" tokenizer using Huggingface's Tokenizers package in Python [8]. A BERT model with one layer of two linear output nodes is then coupled to the tokenizer's generated inputs, creating a head from which binary classification can be calculated. The BertForSequenceClassification model from Huggingface's Transformers Python library is used for this, and it is distill-bertbase-cased. The "bart-large-mnli" model created by Facebook was employed as their zero shot model because to its effectiveness and accessibility on the HuggingFace model portal [8]. .The result emphasizes the point that, despite advances in technology, automatic

classification is not always beneficial for all feedback datasets. The outcomes also demonstrated that a model that is tested and trained on the same dataset works better than a model that is trained on one dataset and then applied to another [8].

The effectiveness of transfer learning (TL) for classifying user reviews was examined by researchers. They trained monolingual and multilingual BERT models and compared their performance to state-of-the-art methods. Henao et al. [21] gathered data from the Google Play Store and the Apple AppStore, totaling 6,406 English app reviews. The second set of data contains 10,364 English tweets, whereas the third set has 15,802 Italian tweets. The Tree-structured Parzen Estimator (TPE) algorithm was used to improve the learning rate. Researchers evaluated their methods using standard metrics such as accuracy, precision, recall, and F1 score [21]. The preprocessing phase has plenty of user comments, including mentions of Twitter accounts prefixed by the letter "@." They replaced these mentions with the phrase "@mention" and then added this word as a token to the BERT tokenizer to avoid the BERT tokenizer splitting them into distinct tokens. To reduce BERT's processing requirements to a minimum, they chose a fixed length of 200 tokens rather than the maximum feasible 512 tokens. Additional special tokens, including the classification (CLS) token, are also introduced to offer the model more information about the comment. The Multilingual BERT model (M-BERT) is trained on two English data sets and then assessed on an Italian data set, and vice versa [21]. The monolingual models outperform the multilingual models substantially. The findings of this study show that M-BERT outperforms cross-lingual transfer learning from Italian to English when trained on English data and then applied to Italian tweets [21].

Considering existing labeled datasets, this paper [9] presents a framework for comparing text embeddings of user feedback. Many future tools for gathering requirement information from user feedback will benefit from these embedding strategies. Topic modeling, averaged word embeddings, word frequency embeddings, and trained text embedding model embeddings are the 4 different types of text embeddings that were analyzed. The LDA, BTM, and Gibbs sampling technique for a Dirichlet Mixture Model(GSDMM) were the three topic modeling approaches that were investigated [9]. Bag-of-words (BOW) and term frequency-inverse document frequency(TF-IDF) was the word frequency embeddings that were tested. The English stop-word list from the NLTK package has been used to test approaches that maintain and eliminate stop words from the text [9]. A uni and bigram model for both BOW and TF-IDF has also been tested using the stop words eliminated model, with a minimum term frequency for terms inside the corpus set. SentenceTransformers' implementations of SBERT, nli-bert-large (S-BERT), and nli-roberta-large (S-RoBERTa), as well as the SentenceTransformers implementation of LaBSE, were employed. LaBSE has been included because it is trained to semantically embed text cross-lingually for over 100 languages, contributing to the complexity of the deep model pre-training methods analyzed [9]. The shown better performance of these pre-trained models across several aspects of user feedback can help shape future tools, especially those that cluster or rank feedback [9].

**The above articles demonstrate the usage of Bert models in the classification of user feedback. The performances of BERT models and conventional ML techniques in the classification of English Tweets were not significantly different, according to the researchers. mBert did not perform well when trained on Italian data [21]. The findings of a study showed that the USE model performed better than the BERT model at grouping feedback with related requirements [9].**

### 5.5.6 Clustering Algorithms

This paper [53]introduces CLAP (Crowd Listener for Release Preparing) which is a comprehensive method for categorizing user reviews based on the information they provide, clustering similar reviews, and automatically prioritizing the clusters of reviews to be applied when planning the next app release. CLAP classifies user reviews using the Weka implementation of the Random Forest machine learning algorithm. To remove negated terms from the reviews, they used the Stanford parser [53]. A set of 200 user reviews was manually collected from five Android apps: Facebook, Twitter, Yahoo Mobile Client, Viber, and Whatsapp. The preprocessing phase included stop words removal, and stemming using the Porter Stemmer and N-grams extraction. DBSCAN, a clustering algorithm that classifies clusters as areas of high element density and assigns elements in low-density regions to singleton clusters, is used to perform review clustering [53]. CLAP uses the Random Forest machine learner to categorize each cluster as high or low priority, with high priority denoting clusters that CLAP suggests for inclusion in the next app release [53].

Researchers' goal in this paper [5] is to automate topic extraction to help with a portion of the process. Researchers retrieve the primary topics discussed in user comments, as well as selected lines that are typical of such issues, from the comments. Requirements Engineers can use this information to change requirements for future releases. The proposed method entails adapting information retrieval techniques, such as topic modeling, and assessing them on a variety of publicly available data sets. The pre-processing of the input data includes Tokenizing, changing the text to lower case, Removing Noise, stop words removal [5]. The sentences were then analyzed to see which topics were related to each sentence. The data sets were then classified using three methods, which included manual classification, KMedian classification with the Jaccard coefficient of similarity, and ASUM classification [5]. Both K-Median/Jaccard and ASUM's results seem good. In comparison to manual classification, the results demonstrate that a good representation of topics may be extracted in a short period of time [5].

In this paper [54] MARK, a keyword-based framework used for semi-automated review analysis has been proposed. MARK enables an analyst to use a set of keywords to describe their interests in one or more mobile apps. For the important activities, they developed and implemented a number of automated, customized techniques, including extracting keywords from raw reviews, ranking them, grouping them based on their semantic similarity, searching for the most relevant reviews to a set of keywords, visualizing their occurrence over time, and reporting any interesting patterns [54].

By using K-mean, MARK can divide a set of keywords into smaller subsets of related terms. In this study, the researchers opted to build a custom filtering algorithm for non-English reviews. They conducted exploratory research to understand more about the characteristics of reviews written in languages other than English. The researchers randomly selected 400 reviews and manually categorized them as either English or non-English reviews. They performed Word stemming using the Porter snowball stemmer and Part-of-Speech tagging. The stemmer divides the review into sentences and then uses the Stanford PoS tagger to identify verbs and nouns in every sentence [54]. For its Review Search function, MARK uses the Vector Space Model. It employs the tf.idf (term frequency-inverse document frequency) to characterize each review as a vector. MARK can assist developers in finding the most meaningful reviews to a set of keywords discovered in the preceding steps. They altered the thresholds of English uni-gram and bi-gram ratios from 0 to 1 with a 0.01 increment in this experiment, then used those thresholds to classify and estimate classification accuracy [54]. Ultimately, the uni-gram and bi-gram thresholds of 0.64 and 0.38, respectively, yielded the greatest accuracy of 86.5 percent [54].

**The above articles have used clustering algorithms namely DBSCAN and Kmean. In an article [53] authors have used DBSCAN algorithm for clustering and random forest to classify each of those clusters as high or low priority. In a paper [54] authors used keyword clustering which resulted in good accuracy.**

### 5.5.7 Random Forest

M. McCord et al. found considerable scope for spammers to infiltrate legitimate users and conducted a study to identify the key features that separate spammers and legitimate users [35]. With user-based and content-based features, M. McCord et al. [35] build a classifier to detect spam data from regular tweets. User-based features are based on a user's relationships like friends, and content-based features are decided based on the tweet's length [35]. To perform the study to detect spam tweets, M. McCord et al. collected 100 most recent tweets from 1000 random Twitter users and labeled them manually if they were spam or not spam [35]. With the labeled data, M. McCord et al. [35] used classification algorithms like Random forests, Naive Bayesian, Support Vector Machine using SMO scheme implemented in WEKA tool, and K-nearest neighbours using WEKA tool as IBK classifier [35]. These classifiers are compared using three metrics as Precision, Recall, and F-measure [35].

**The authors of this study [35]used four classification algorithms for spam detection: random forest, SVM, Naive Bayes, and K-nearest neighbors. The results of this study show that the random forest classifier has outperformed other classifiers in detecting spammers [35].**

## 5.6 Conclusion

From the research conducted by many researchers to automate the requirements-gathering process from users' feedback, there are two main concepts, clustering, and classifiers. The authors performed preprocessing on the input with classifiers and clustering models as the core of automation. Preprocessing includes NLP (Natural Language Processing) techniques like tokenizing, lemmatizing, etc., and other techniques to clean data, removing or filling null values. Although there are only a few working models that can automate the requirements-gathering process using clustering, classification models showed excellent results and are used by many researchers to automate the requirements-gathering process. There is no manual work or prerequisite needed to perform clustering; clustering can be performed on ready data (data after cleaning and tokenized) directly, but for classification, one should have the data labeled manually by the researcher's party since it is a supervised learning model. The results of our literature review show that SVM has performed well in most of the research studies [24] [23] [36] compared to the other classifiers which are why we chose Support Vector Machines(SVM) algorithm in our case study.

# Chapter 6

# Case Study

Our main objective is to use a natural language processing model to classify requirements automatically from users' feedback. We have a case study prepared to perform the machine learning methods and techniques from the SLR Literature review. The case study is Telia's Twitter feedback given by its users. The reason for selecting Telia as our case study for this thesis is that the data has many examples of tweets in both English and Swedish.

## 6.1 Data

Twitter tweets are the source of this paper, and the data is a set of small messages ranging from 120-140 characters. These are user-generated feedback on telia company related to their services, application, and web app, and this data is in Swedish and English.

The extracted data is taken in an Excel sheet. The extracted information has ten columns: "source", "app", "user name", "rating", "title", "review", "at", "isEdited", "reply content", and "repliedAt". The "source" column has the source of information taken; the "app" column is the application on which the feedback is given; "user name" is the name of the user who gave his feedback; the "rating" column has the rating given by the user; the "Title" is left as an empty column; "review" column has the feedback given by the users; "at' column shows the time of feedback; "isEdited" is a column with True and False values; "reply content" column has the reply given by the company and "repliedAt" column has the time of reply given by the company.

## 6.2 Data-Preprocessing

The data extracted from Twitter is not structured well and has many empty values(null values); columns are merged as one column separated with commas, and each row is separated with '; ;'. This data cannot be used directly in automation models. To make the data extracted directly from Twitter ready for automation, we have performed a set of data cleaning and pre-processing techniques.

### 6.2.1 Structuring the data

The Twitter data we have has two types of structures. First, is its data separated through columns in an excel sheet, and second, the data in the row is separated with

commas in one column. Each alternate row is empty with ';;' in its first column. Each row ended with ';;'.

To Deal with such structureless data, first, we have converted the whole data into a comma-separated values text file.

Listing 6.1: code to remove comma and semicolon

```
f = open("comma separated file path//file1.txt","r",errors='ignore')
f1= open("tab spaced file path//file2.txt","a")
for x in f:
        txt=x.replace(",","\t")
        txt=txt.replace(";","")
        f1.write(txt)

f1.close()
f.close()
```

We programmed a python code to convert commas to tabs to have a single structured data. We used the file open method in python to open two files, one is to read the comma-separated, and the other is a blank file to append the data by replacing the commas with tab spaces. The open method takes mainly two parameters, one is the source of the file, and the other is a mode that specifies the mode in which the file is opened, i.e., read mode, append mode, and write mode. Here, we opened the first file "f" in read mode i.e., "r" which allows only read functionality on the opened file, and the second file with append mode allowing both read and append functionalities.

After opening the files, we used a loop variable "x" that will read each character from the file "f" and append it to the second file "f1". When the value of "x" is "," we used the replace() method in strings to replace "," with tab space, and ";" with " ".

We used Google Sheets to convert tab-separated values into an excel sheet or a data frame.

### 6.2.2 Extracting Data

We have ten columns or ten features in the data set. We considered only one feature which is reviews provided by the users. To get the required features, we used the pandas library in python and extracted the data in a separate CSV file that can be used as a data frame for performing automation.

## 6.3 Approach to automate classification

The core model we decided to implement on the data set to automate the classification of requirements is an SVM classifier. We selected the SVM model by comparing the results of the SLR study conducted to answer RQ1.

### 6.3.1 Classifier

It is a supervised learning method. We got the data labeled to use classifiers in automation, and we decided to use two ways of using classifiers.

- A binary classifier for each class. We want to test different classifiers and identify the best classifier. So that we will run the test tweet through the selected classifiers, and based on the results, we will classify the feedbacks.
- A multiclass classifier for all classes. We decided to run different multiclass classifiers on the labeled data. By comparing the mean accuracies of the classifiers, we decided to choose one classifier to automate the classification of tweets into possible Requirements.

## 6.4 Tackling Multi-Lingual data

From our results from SLR and browsing, we found Libraries that can deal with Swedish vocabulary. SpaCy, mBERT, and XML-R are the libraries that can lemmatize and tokenize Swedish data. In this paper, we will be using the SpaCy library to deal with Swedish data. The spaCy library is more reliable technology compared to the other mentioned libraries because of its documentation and its support for python coding and it mainly contains both Swedish and E nglish vocabulary. Since the data is in both Swedish and English data, we built two models for Swedish data and English data and pass English data to the English model and Swedish data to the Swedish model.

The data we have is in both Swedish and English. Since we are using two different models, to train both the models with enough data we translated the whole data into completely Swedish and completely English data using googletrans API.

## 6.5 Implementing Classifiers

In this module, we built supervised machine-learning models. The data that we have is not ready for supervised machine learning models. So the first step is to have labeled data.

### 6.5.1 Data Labeling

After learning about the tweets, we identified eight categories

- Authentication issue
- Interface issue
- Bugs and update issues
- Feature required
- Platform issue

- Reliability issues
- Service issue
- Spam

**Authentication issue:**
The data is categorized under authentication issue if the user reports any login issues.

**Interface issue:**
The data is categorized under interface issue if the user reports any difficulty dealing with the user interface.

**Bugs and update issues:**
The data is categorized under bugs and update issues if the user reports the difficulty faced after updates or any undesirable functionalities in the application.

**Feature requested:**
The data is categorized under Feature requested if the user suggests or requests any feature for the application.

**Platform issue:**
The data is categorized under Platform issue if the user finds any difficulty mentioning the OS or device while using the application.

**Reliability issues:**
The data is categorized under Reliability issues if the user reports not working applications or any reliability issues.

**Service issue:**
The data is categorized under Service issue if the user reports bad service, including customer services, payment services, etc.

**Spam:**
The data that does not come under any of the above categories are considered spam.

### 6.5.2 Requirements from User Feedback

In this case study, we selected the labels after carefully reviewing all feedback. Depending on the feedback given, we have prioritized all the negative feedback. A user cannot specify a requirement instead, he will report it as an issue. However, reported issues can be a valuable source of information for identifying requirements. By analyzing the problems and difficulties that users face with a product, one can identify areas for improvement and new features that should be included in future versions of the product. This process can help to ensure that the requirements for the product align with the needs and expectations of its users. [13]

### 6.5.3 Data-Labeling Credibility

We manually labeled the data, and then to confirm the correctness of the labeling, we took the help of a student group who reviewed the labeling. To check the credibility of the labeled data, we randomly took 20% of unlabeled tweets and gave them to our mentor Dr. Krzysztof Wnuk, an assistant professor at the Software Engineering Research Group (SERL) of Blekinge Institute of Technology, Sweden. We took the labeled data from Dr. Krzysztof Wunk and compared the labeling with our data, which matched correctly with a 100% match result.

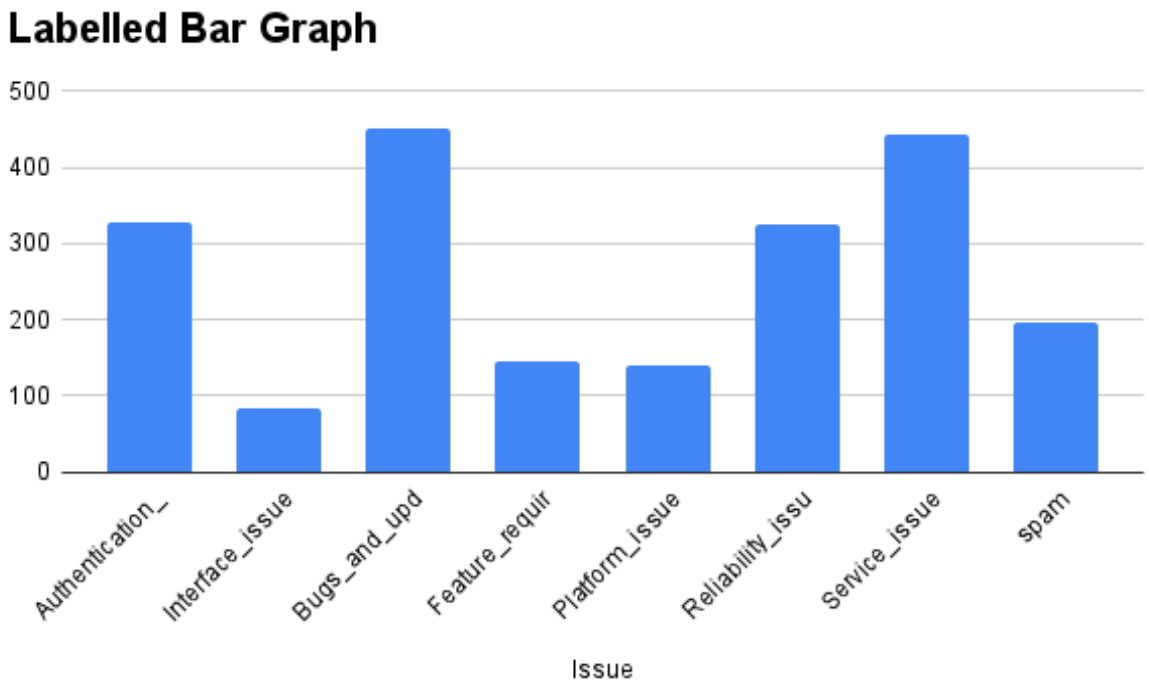


Figure 6.1: Labeled Data Bar-graph

Figure 6.1 represents a bar graph showing the distribution of data among the seven categories.

### 6.5.4 Case study Model

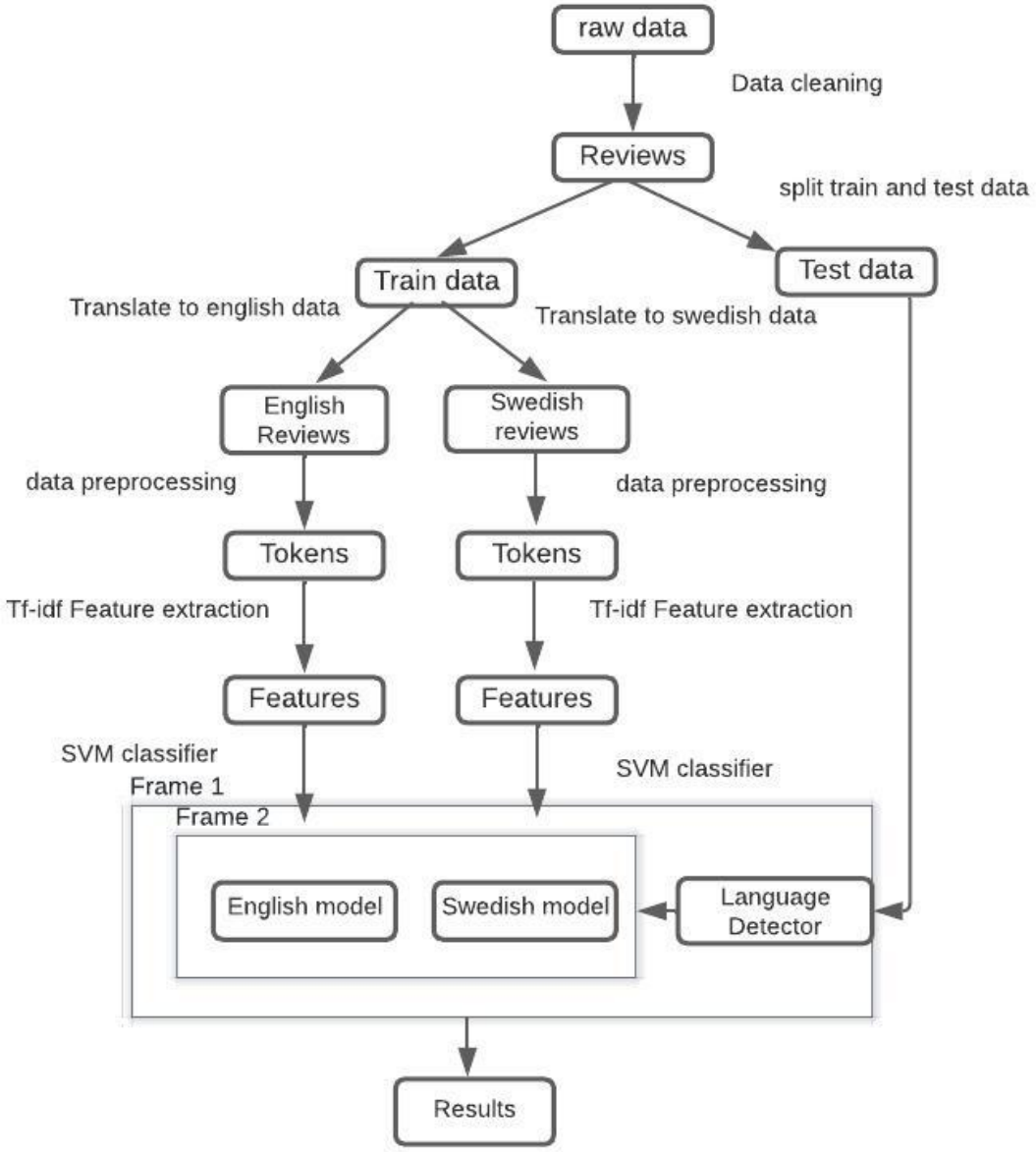


Figure 6.2: Model design for our case study

### 6.5.5 Data Pre-processing

The pre-processing phase is divided into 3- steps.

- Removing NaN and empty fields
- Tokenization
- Feature Extraction

**Removing NaN :** The data present has empty fields which are considered by the system as either null or NaN value. To remove NaN values we used pandas fillna(0) dunction which replaces null values with 0.

**Tokenization :** For English model, we used nltk library to download 'stopwords' to remove stop words from the reviews and used regularexpressios to remove puncyiation marks. For Swedish model, we used spacy library to load swedish vocabulary i.e, "sv_core_news_lg". Using the vocabulary provided by spacy we removed stop words and punctuations by identifying them using is_stop and is_punct.

**Feature Extraction :** We employed Tfidf-Vectorizer technique to extract features from generated tokens in both Swedish and English languages.

### 6.5.6 Implementing Binary Classifier for each Category and Multi-class Classification

Listing 6.2: Python code for tokenization in Swedish

```
import spacy
import re

nlp = spacy.load("sv_core_news_lg")
words=[]
for i in range(0,2109):
  doc = nlp(str(df['reviews'][i]).replace(r'\d+',''))
  word = ' '.join([token.lemma_.strip() for token in doc
          if not token.is_stop and not token.is_punct])
  words.append(word)
```

We decided to build separate binary classification models for each class and a multiclass classifier for all classes. We first used the spacy library to get Swedish vocabulary using spacy.load("sv_core_news_lg"). by assigning the vocabulary to a variable 'nlp', we used the string replace method to replace digits with spaces from the reviews with nlp(str(df['reviews'][i]).replace(r'd+',' ')), here df is the data frame with reviews and labels; 'reviews' is the column names and i is to iterate each row in a for-loop.

Listing 6.3: Python code for tokenization in English

```
import spacy
import re

nlp = spacy.load("en_core_web_lg")
words=[]
for i in range(0,2109):
  doc = nlp(str(df['reviews'][i]).replace(r'\d+',''))
  word = ' '.join([token.lemma_.strip() for token in doc
          if not token.is_stop and not token.is_punct])
  words.append(word)
```

Then we took the cleaned reviews, and removed stop words and punctuations by identifying them using token.is_stop and token.is_punct from the spacy library,

where the token is a loop variable to read every word from the review. After removing all punctuations and stop words we lemmatized the review using tokenlemma_ from the spacy library, and appended the review to a list words[], this words list we use later to extract the features.

Listing 6.4: Python code for pipeline

```
from sklearn.pipeline import Pipeline
from sklearn.svm import LinearSVC
from sklearn.feature_extraction.text import TfidfVectorizer
text_clf = Pipeline([('tfidf',TfidfVectorizer())
                     ,('svm',LinearSVC())])
```

In the next step, we created a pipeline that performs feature extraction and classification. We used the Pipeline library imported from sklearn.pipeline to create our pipeline. We used the tf-idf feature extraction technique to extract features and the Linear-SVM classifier to classify the model. To perform tf-idf feature extraction, we used TfidfVectorization imported from sklearn.feature_extraction.text, and to use linear svm classifier, we used LinearSVC imported from sklearn.svm. We included both the TF-IDF feature extraction technique and Linear SVM classifier model in the pipeline using Pipeline([('tfidf', TfidfVectorizer()),('svm', LinearSVC())]), we are passing a list containing tfidf feature extractor followed with an svm classifier. In the list, we added a feature extractor and a classifier as two tuples. The first element of the tuple is the name and the second one is the method, here 'tfidf' is the name of the feature extractor and TfidfVectorizer() is the method that implements tf-idf feature extraction, similarly 'svm' is the name for the classifier and LinearSVC() is the method that we use for classification.

We loaded all the tokens into variable X which is X=words, for code convenience. Now we have tokenized reviews 'X' and the pipeline 'text_clf' ready.

Listing 6.5: Python code for binary classifier model

```
from sklearn.model_selection import train_test_split

for i in range(1,len(tr)):
  y=tr.iloc[:,i].values
  X_train, X_test, y_train, y_test =
        train_test_split(X, y, test_size = 0.20, random_state = 0)
  text_clf.fit(X_train,y_train)
  predictions = text_clf.predict(X_test)
```

For the binary classifier on each class, we used a for loop to run on each class. So, now the 'y' will be a loop variable that represents each class in the data frame 'tr'. To train our model we split the data such that 80% of the data is used for training and 20% of the data is used for testing. To split the data we used train_test_split imported from sklearn.model_selection. The parameters that we pass to the train_test_split are: X- tokenized reviews, y- class labeled data, test_size - the value represents the test data size, random_state- to control the shuffling process, if given a state value,

the shuffling will be constant to every split if we use the same randam_state value. The train_test_split method returns a list, in our case, a list with four elements, reviews train and test data, and class labels train and test data. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0) splits the tokenized reviews and labeled column into X_train- reviews to train the model, X_test- reviews to test the model, y_train- labeled data to train the model, and y_ test- labeled data to test the model. Now that we have our train and test data ready, we pass the train data through the pipeline 'text_clf' that we constructed before using the fit() method with X_train and y_train as parameters. With the help of the pipeline fit() method, we will complete the model training and get the predictions to the X_test data using predict() method in the pipeline which takes one parameter which is X_test- test reviews data. We saved the predictions in the 'predictions' variable. We will use 'predictions' later to evaluate the model.

# Chapter 7

# Results and Analysis

In this module, we present our findings from this thesis, results obtained from SRL, and Case Study. Finally, answer the research questions by analyzing the results we got from our study.

## 7.1 SLR Results

Using the snowball technique, we found 29 papers with inclusion criteria ((IC1 or IC3) and IC2) where
**IC1:** About multilingual machine learning
**IC2:** About machine learning for analyzing user feedback (any feedback from social media or general) But EMPIRICAL so real data
**IC3:** Mining Twitter (because it is a short text maximum of 160 characters, so the feedback is brief)

**RQ1: What is the state of the literature in analyzing user feedback for requirements engineering?**
The State of the art for analyzing user feedback for automatic requirements engineering is discussed in Chapter 5 SLR and concluded the study in the following sub-sections

### 7.1.1 State of the literature in analyzing user feedback for requirements engineering

There are two key concepts that emerged from the study undertaken by various researchers to automate the requirements-gathering process based on user feedback: clustering and classifiers. As the foundation of automation, the authors used classifiers and clustering models to preprocess the data. Preprocessing covers NLP (Natural Language Processing) techniques such as tokenizing, lemmatizing, and other data cleaning techniques such as deleting or filling null values. Although there are just a few functioning models that can automate the requirements-gathering process using clustering, classification models produced good results and are employed by numerous researchers.No manual effort or requirement is required to do clustering; clustering can be performed immediately on data after cleaning, and tokenization, whereas classification requires the data to be labeled manually by the researcher's side because it is a supervised learning model.

## 7.2 Case Study Results

**RQ2: What methods can be used to automate the analysis of multilingual feedback?**
The case study of this thesis is Telia's Twitter feedback. We built a multi-lingual classification application combining both Swedish and English models. The results are presented and discussed below.

### 7.2.1 Case Study Data

The case study data we used is Twitter feedback on Telia company. We got this data from our mentor Prof. Krzysztof Wnuk. The given data set is in both Swedish and English languages. The total number of multi-lingual data present in the given Twitter data set is 2110. The number of English tweets in the dataset is 853, and the number of Swedish tweets is 1257.

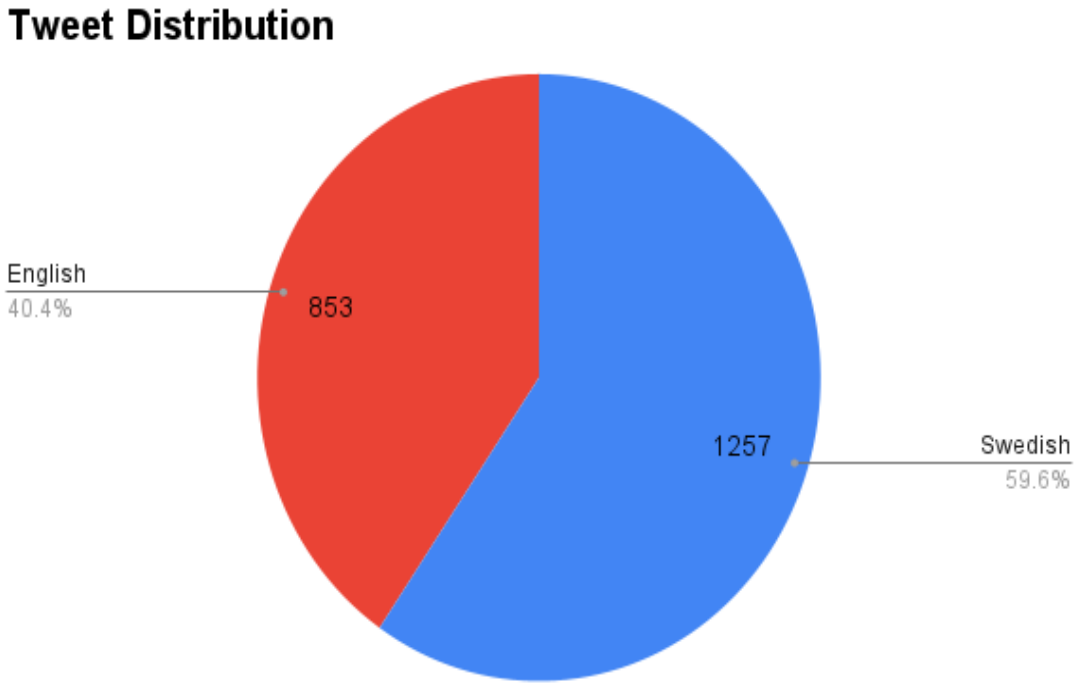


Figure 7.1: Tweet Distribution

### 7.2.2 Score and Result metrics

**Confusion Matrix:** [55] The confusion matrix is a prominent metric for solving classification problems and can be used for binary and multiclass classification problems. The confusion matrix is made up of four main properties (numbers) that are utilized to define the classifier's measuring metrics. These four figures are:

- True positive (TP):
  A test result that correctly indicates the presence of a condition or characteristic
- true negative (TN):
  A test result that correctly indicates the absence of a condition or characteristic
- false positive (FP):
  A test result that wrongly indicates that a particular condition or attribute is present
- false negative (FN):
  A test result that wrongly indicates that a particular condition or attribute is absent

**Accuracy:**
A standard way to assess a classifier's performance in machine learning is to evaluate its accuracy. It is referred to as the ratio of correctly classified elements to the total number of elements [28].

$$accuracy = \frac{true\ positives + true\ negatives}{true\ positives + true\ negatives + false\ negatives + false\ positives}$$

Figure 7.2: Formula for accuracy [28]

**Recall:**
Recall depends on how effectively the model identifies all the positives. True positive rate is another name for recall [28].

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

Figure 7.3: Formula for recall [28]

**Precision:**
A measure of precision shows you what percentage of the total positives the model predicts are true positives.

The definitions of recall and precision demonstrate how closely they are related to each other [28].

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

Figure 7.4: Formula for precision [28]

**F1-score:**
A measurement that combines recall and precision is the F1 score. F1 can be used to evaluate how well our models manage the trade-off between precision and recall [28].

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figure 7.5: Formula for F1-score [28]

### 7.2.3 Multi-lingual Problem

The data presented in this paper is Twitter feedback data on Telia company. This Twitter data is in two languages English and Swedish. We build a model that will process Swedish data and English data with the help of the vocabulary provided by Spacy. The vocabulary for Swedish is "sv_core_news_lg" and the vocabulary for English is "en". These vocabularies are downloaded to our python code using spacy.cli.download(vocabulary). The data we have is divided between Swedish and English languages. The data we have is not sufficient to properly train the model. So, to train the model with more data we split the model into two models one that understands English vocabulary and the other that processes Swedish vocabulary. We converted the whole data into English language and Swedish language, to feed the respective models with more data. Note: the preprocessing, feature extraction, and classifier model used in the two models are the same except for the vocabulary. In testing, we used TextBlob imported from textblob to identify the language. Depending on the language of the test data, the respective model is selected to get the score.

### 7.2.4 Classifiers Result

The Twitter data is labeled manually to perform classification. The classification is performed in two ways

- Multiple Binary Classifiers
- Multi-class Classifier

We have used two types of classifiers to compare the results and decide which type of classifier is more accurately classifying the requirements from users' feedback.

#### 7.2.4.1 Multiple Binary Classifiers

To calculate the score of the models we used a confusion_matrix and classification_report imported from sklearn.metrics. We used a confusion matrix to know the true positive, true negative, false positive, and false negative; A classification report

to know precision, recall, f1-score, accuracy, and macro average which is an average of all results metrics. The binary classifiers -

- SVM (Support Vector Machine) for English data
- SVM (Support Vector Machine) for Swedish data

The results of the classifiers are validated using accuracy, precision, and recall.

| **S.No** | **Class Name** | **Accuracy** | **Precision** | **Recall** |
|---|---|---|---|---|
| 1 | Authentication issues | 0.94 | 0.92 | 0.89 |
| 2 | Interface issues | 0.98 | 0.97 | 0.65 |
| 3 | Bugs and update issue | 0.86 | 0.80 | 0.73 |
| 4 | Feature required | 0.91 | 0.72 | 0.60 |
| 5 | Platform issues | 0.98 | 0.97 | 0.80 |
| 6 | Reliability issues | 0.88 | 0.77 | 0.71 |
| 7 | Service issues | 0.81 | 0.73 | 0.70 |

Table 7.1: Score for SVM (Support Vector Machine) for English data

| S.No | Class Name | Accuracy | Precision | Recall |
|---|---|---|---|---|
| 1 | Authentication issues | 0.94 | 0.91 | 0.89 |
| 2 | Interface issues | 0.96 | 0.73 | 0.60 |
| 3 | Bugs and update issue | 0.85 | 0.76 | 0.70 |
| 4 | Feature required | 0.93 | 0.66 | 0.65 |
| 5 | Platform issues | 0.98 | 0.92 | 0.75 |
| 6 | Reliability issues | 0.87 | 0.75 | 0.70 |
| 7 | Service issues | 0.81 | 0.74 | 0.71 |

Table 7.2: Score for SVM (Support Vector Machine) for Swedish data

#### 7.2.4.2 Multi-class Classifier

Similar to the binary classifier model, we used a confusion_matrix and classification_report imported from sklearn.metrics to calculate the score of the multi-class classifier model. We used a confusion matrix to know the true positive, true negative, false positive, and false negative; A classification report to know precision, recall, f1-score, accuracy, and macro average, which is an average of all results metrics.

**Score for SVM (Support Vector Machine) :**
To evaluate the complete model that classifier both Swedish and English tweets we combined two models (English and Swedish models) and passed the multi-lingual test data to these models based on the language of the test data with the help of a Language Detector. The average accuracy, weighted average precision, and weighted average recall values are:

- Accuracy: 0.71
- Precision: 0.67
- Recall: 0.68

# Chapter 8

# Discussion

In this section, we will summarise the research questions, methods followed to answer the questions, and results that answer the research questions. We discuss the issues that disagree with the research and how this thesis can contribute to research and practice.

RQ1: What is the state of the literature in analyzing multilingual user feedback for requirements engineering?
To answer this research question we performed SLR with Snowballing approach. In the study, we found 29 papers explaining the state-of-the-art technology to automate requirements gathering and multi-lingual requirements classification. As part of the Systematic Literature Review, we have explored different machine-learning and deep-learning techniques that can help classify short messages automatically. We also learned about tools like ALERTme, NIRMAL, SIMBA, Weka, AR-Miner, Doccano, CLAP, etc., and how they help achieve automation. These findings are clearly presented in Chapter 5- SLR. From the SLR study, we found 8 papers that have used the SVM classification model. Since the SVM classification model is the most used classification model among Naive Bayes, Decision Tree, Random Forest, Neural Networks, and Logistic Regression models, we decided to use the SVM classification model to classify the requirements. To deal with NLP, BERT, NLTK, and spaCy are the technologies used in the selected 29 research papers, which support the python language. We used the spaCy library to process NLP data because it provides Multi-Lingual support, and clear and easy to understand documentation [34]. The feature extraction technique we implemented from the SLR study is Tf-idf vectorization. Although Bag of words is suggested in the SLR study, the results are more accurate with Tf-idf compared with the Bag of words [39] [24]. With the above suggestions and inputs from our SLR study we conducted the case study to answer the RQ2.

RQ2: What method can be used to automate the analysis of multilingual feedback?
By getting to know the current state-of-the-art technology for automating multilingual text classification using a Systematic Literature Review, we answer the research question. We chose the Case study methodology on the Twitter feedback on Telia company. The tweets are in both Swedish and English language. There are 2110 reviews. We manually reviewed the data and found eight categories of requirements while going through each of the reviews manually. We manually labeled the tweets and verified them by randomly selecting 20% of the tweets and getting them

labeled manually by our mentor Krzysztof Wnuk, then compared our labeled data with Krzysztof Wnuk's labeled data. The labeling accuracy is 100 percent between us and the professor. To implement NLP on multi-lingual data we used the spaCy library since it supports python language. It provides feature extraction techniques like count-vector and tf-idf vectorization. spaCy has clear and structured documentation. We created a pipeline with tf-idf feature vectorization and SVM so the tokens pass through the tf-idf first and then the extracted features are passed through the SVM. We built two types of classifying models, one to perform binary classification on each classification or column and the other to perform multi-class classification on all classes at once. We tried to build a model that can process Swedish and English data, but we did not find a vocabulary that includes the Swedish and English languages. So we developed a model by combining two svm models, one trained with Swedish vocabulary and the other trained with English vocabulary. Then tested the data by passing the data through the model with two separate classifiers and the results are presented in Chapter 7. Based on the accuracy score, we understand that the data distribution between labels is unbalanced. This imbalance in data might cause overfitting in the model. So, we implemented an upsampling method to solve the overfitting problem, but the accuracy and precision score did not improve. So, we did not include it in the case study.

### 8.0.1 Implementations for research and practice

In the software life-cycle, requirements gathering is the most important and likely to have more errors. Requirements are taken from end users and customers. These requirements then work like a map in the development cycle. One can often miss some requirements or not get the right source to collect the requirements. We chose to address the gaps in requirements engineering. The thesis's main goal is to identify requirements from users' feedback in Swedish and English languages. We developed a model that can understand Swedish and English data and then classify them into possible requirements. This model can be more helpful for Swedish-based software companies in achieving more clear requirements for their products. The model we created is unique, and with the present market in NLP, our model has good scope for research and implementation.

## 8.1 Validity threats

In this section, we will discuss the threats involved in this Thesis paper and how to mitigate them.

### 8.1.1 Internal Validity

Internal validity indicates the correctness of the research. The data we received from our mentor Krzysztof Wnuk is valid and appropriate. However, the data, when reviewed manually we found seven features, and the data is not sufficient to get exact results. The labeling is done manually by us, but since we do not know Swedish, we translate the Swedish data to English data to label the complete dataset. This translation might cause the loss of data. Provided with more data the model can perform much better. In this paper, we performed a case study, implementing methods taken from our SLR study. The developed models in this thesis have performed well and can perform better with more data.

### 8.1.2 External Validity

External validity indicates the extent to which the results from the study can be used or generalized. The model developed as part of the study can be used on any labeled data in Swedish or English or both Swedish and English data. The model can be further improved based on the data used. With the help of the vocabulary in spacy, the model scope can be extended to different languages.

### 8.1.3 Construct Validity

It's important to take construct validity into account when collecting data. As a result, construct validity describes how well research accomplishes its stated objectives, or how well a process results in an accurate assessment of actuality. The construct validity of SLR is heavily influenced by the validity of the papers included in the snowballing process. We generated a search string and three requirements for inclusion criteria to increase validity. Additionally, our supervisor must review all of the articles produced from the start set to the last iteration before any of the articles being added in the SLR. As part of our case study, we have chosen the SVM algorithm for the classification process. Construction validity problems may occur if the data is not formulated properly.

### 8.1.4 Conclusion Validity

Conclusion validity is the degree to which the conclusion we acquire is reasonable or convincing [6]. There can be a conclusion validity threat if a process is not followed correctly. We should read an article several times if it is poorly written so that we can better understand its purpose.To mitigate this risk, we must try to understand what the author is trying to express in that particular article. A threat to the conclusion may result from improper process execution. The data should be

thoroughly extracted to reduce this threat, as it is important to answer the research question based on this data.

### 8.1.5 Risk Managements

| S.No | Research Question | Risk | Mitigation |
|---|---|---|---|
| 1 | RQ1, RQ2 | There can be a risk of having very few papers concerning models or methods to analyze requirements from multi-lingual data-driven data | If there are few papers on analyzing user feedback for requirements engineering, increasing the search scope towards methods that can classify data-driven data based on the keywords or searching for articles concerning automation models over multi-lingual data can mitigate the risk of not finding enough papers for our literature review. |
| 2 | RQ2 | We have selected the case study methodology to answer RQ2. By the nature of methodology, a case study always poses a risk for generalization. The results obtained from a case study cannot be the generalized result used for other similar problems. [19] | We will present all models and methods that we found and used for our case and explain the complete process for finding the solution in a generalized way so that readers can follow the procedure if they are working on a similar problem. |
| 3 | RQ1, RQ2 | Falling behind the schedule and deadlines | Estimating the work and time at the initial stage of the thesis is crucial. Dividing the work into small tasks and achieving them will in time maintain the momentum of work and having regular meetings with the supervisor can help in not falling behind the schedule. |

Table 8.1: Limitations and Risk Managements

# Chapter 9

# Conclusions and Future Work

## 9.1 Conclusion

In this thesis, we developed an NLP model that extracts requirements from users' feedback on Telia on Twitter. This model can understand Swedish and English tweets. This model can help companies to improve their requirements gathering, mainly in Sweden, because the model has Swedish vocabulary.

To answer RQ1, we conducted an SLR study using the snowballing approach. In this study, we found 75 papers after snowballing, which were reduced to 29 papers using an inclusion criteria condition (IC2 and (IC1 or IC3)). From the study, we learned about NLP feature extraction techniques like Tf-idf, BOW, term-frequency, etc. The methods used for automation of requirements extraction are sentiment analysis, SVM, Naive-Bayes, LDA, Bert, Clustering, and Random forest. With the help of the SLR study, we understand the state-of-the-art research on requirements gathered from multilingual tweets.

To answer RQ2, The case study we performed is on the Twitter feedback on Telia company. The tweets are in Swedish and English language. To automatically classify the multilingual data into requirements, we built a model with two svm classifiers, for Swedish and English vocabulary, and Tf-idf feature extractions to pre-process text data. To have a clear understanding of the model and its accuracy, we performed classification on every class and all classes at once. With binary classification on each class, the accuracies for Authentication issues, Interface issues, Bugs, and update the issue, Feature required, Platform issues, Reliability issues, and Service issues are 0.94, 0.98, 0.86, 0.91, 0.98, 0.88, 0.81 respectively for English data, and 0.94, 0.96, 0.85, 0.93, 0.98, 0.87, 0.81 for Swedish data. For multi-class classification, we did not separate the test data into Swedish and English data. The accuracy i= 0.71, precision = 0.67, and recall = 0.68. The results state that the performance of the model is not good. As we discussed in the previous chapter the cause for such a performance could be an imbalanced distribution of labels across the reviews.

## 9.2 Future Work

- The SLR study we performed can be improved by expanding the scope of the research. There are many papers about building NLP models on Swedish data and how to build our vocabulary using the given data. Due to time constraints, we did not include those papers in our research study.
- The models for English and Swedish data are built using the Python Spacy library. Some more technologies and tools can support Swedish vocabulary like mBERT, TextBlob, etc. We did not experiment with other libraries or tools mentioned before, but one can use them to check the results.
- The dataset we have has 1257 Swedish tweets and 853 English tweets. By changing the proportions of Swedish and English data, one can study how the results differ and find the certain proportion of Swedish and English data at which the model provides high accuracy score.

# References


[1] "https://ebbot.ai/open-source-resources-for-nlp-tasks-in-swedish/."

[2] S. Asiri, "Machine learning classifiers." [Online]. Available: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

[3] M. Binkhonain and L. Zhao, "A review of machine learning algorithms for identification and classification of non-functional requirements," *Expert Systems with Applications: X*, vol. 1, p. 100001, 2019.

[4] S. Bird, "Nltk: the natural language toolkit," in *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, 2006, pp. 69–72.

[5] L. V. G. Carreño and K. Winbladh, "Analysis of user comments: An approach for software requirements evolution," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 582–591.

[6] Conjointly, "Threats to conclusion validity." [Online]. Available: https://conjointly.com/kb/conclusion-validity-threats/

[7] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," 2019.

[8] P. Devine, Y. S. Koh, and K. Blincoe, "Evaluating software user feedback classifiers on unseen apps, datasets, and metadata," *arXiv preprint arXiv:2112.13497*, 2021.

[9] ——, "Evaluating unsupervised text embeddings on software user feedback," in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2021, pp. 87–95.

[10] V. T. Dhinakaran, R. Pulle, N. Ajmeri, and P. K. Murukannaiah, "App review analysis via active learning: Reducing supervision effort without compromising classification accuracy," in *2018 IEEE 26th International Requirements Engineering Conference (RE)*, 2018, pp. 170–181.

[11] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall, "What would users change in my app? summarizing app reviews for recommending software changes," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 499–510.

[12] A. M. Ebrahimi and A. A. Barforoush, "Preprocessing role in analyzing tweets towards requirement engineering," in *2019 27th Iranian Conference on Electrical Engineering (ICEE)*. IEEE, 2019, pp. 1905–1911.

[13] M. Fischer, M. Pinzger, and H. Gall, "Analyzing and relating bug report data for feature tracking," in *WCRE*, vol. 3, 2003, p. 90.

[14] B. Flyvbjerg, "Case study," *The Sage handbook of qualitative research*, vol. 4, pp. 301–316, 2011.

[15] G. for Geeks, "Classifying data using support vector machines(svms) in python." [Online]. Available: https://www.geeksforgeeks.org/classifying-data-using-support-vector-machinessvms-in-python/?ref=rp

[16] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 1276–1284.

[17] E. C. Groen, N. Seyff, R. Ali, F. Dalpiaz, J. Doerr, E. Guzman, M. Hosseini, J. Marco, M. Oriol, A. Perini *et al.*, "The crowd in requirements engineering: The landscape and challenges," *IEEE software*, vol. 34, no. 2, pp. 44–52, 2017.

[18] E. Guzman, R. Alkadhi, and N. Seyff, "A needle in a haystack: What do twitter users say about software?" in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, 2016, pp. 96–105.

[19] E. Guzman, M. Ibrahim, and M. Glinz, "A little bird told me: Mining tweets for requirements and software evolution," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 11–20.

[20] E. Guzman and W. Maalej, "How do users like this feature? a fine grained sentiment analysis of app reviews," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, 2014, pp. 153–162.

[21] P. R. Henao, J. Fischbach, D. Spies, J. Frattini, and A. Vogelsang, "Transfer learning for mining feature requests and bug reports from tweets and app store reviews," in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, 2021, pp. 80–86.

[22] M. Hosseini, K. T. Phalp, J. Taylor, and R. Ali, "Towards crowdsourcing for requirements engineering," 2014.

[23] T. Iqbal, M. Khan, K. Taveter, and N. Seyff, "Mining reddit as a new source for software requirements," in *2021 IEEE 29th International Requirements Engineering Conference (RE)*. IEEE, 2021, pp. 128–138.

[24] N. Jha and A. Mahmoud, "Using frame semantics for classifying and summarizing application store reviews," *Empirical Software Engineering*, vol. 23, no. 6, pp. 3734–3767, 2018.

[25] C. Khanna, "Text pre-processing: Stop words removal using different libraries." [Online]. Available: https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a

[26] G. Koelsch, "Requirements writing for system engineering." [Online]. Available: https://learning.oreilly.com/library/view/requirements-writing-for/9781484220993/A420090_1_En_1_Chapter.html

[27] ——, "Requirements writing for system engineering." [Online]. Available: https://learning.oreilly.com/library/view/requirements-writing-for/9781484220993/A420090_1_En_1_Chapter.html

[28] labelf, "What is accuracy, precision, recall and f1 score?" [Online]. Available: https://www.labelf.ai/blog/what-is-accuracy-precision-recall-and-f1-score

[29] C. Li, L. Huang, J. Ge, B. Luo, and V. Ng, "Automatically classifying user requests in crowdsourcing requirements engineering," *Journal of Systems and Software*, vol. 138, pp. 108–123, 2018.

[30] S. Lim, A. Henriksson, and J. Zdravkovic, "Data-driven requirements elicitation: A systematic literature review," *SN Computer Science*, vol. 2, no. 1, pp. 1–35, 2021.

[31] M. Lu and P. 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*, 2017, pp. 344–353.

[32] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Engineering*, vol. 21, no. 3, pp. 311–331, 2016.

[33] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe, "Toward data-driven requirements engineering," *IEEE software*, vol. 33, no. 1, pp. 48–54, 2015.

[34] D. Martens and W. Maalej, "Extracting and analyzing context information in user-support conversations on twitter," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019, pp. 131–141.

[35] M. Mccord and M. Chuah, "Spam detection on twitter using traditional classifiers," in *international conference on Autonomic and trusted computing*. Springer, 2011, pp. 175–186.

[36] S. McIlroy, N. Ali, H. Khalid, and A. E Hassan, "Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1067–1106, 2016.

[37] M. E. Mezouar, F. Zhang, and Y. Zou, "Are tweets useful in the bug fixing process? an empirical study on firefox and chrome," *Empirical Software Engineering*, vol. 23, no. 3, pp. 1704–1742, 2018.

[38] M. Nayebi, H. Cho, and G. Ruhe, "App store mining is not enough for app improvement," *Empirical Software Engineering*, vol. 23, no. 5, pp. 2764–2794, 2018.

[39] E. Oehri and E. Guzman, "Same same but different: Finding similar user feedback across multiple platforms and languages," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 2020, pp. 44–54.

[40] F. Palomba, P. Salza, A. Ciurumelea, S. Panichella, H. Gall, F. Ferrucci, and A. De Lucia, "Recommending and localizing change requests for mobile apps based on user reviews," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 2017, pp. 106–117.

[41] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in *2015 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 2015, pp. 281–290.

[42] 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)*. IEEE, 2012, pp. 596–599.

[43] L. Ramadhan, "Tf-idf simplified." [Online]. Available: https://towardsdatascience.com/tf-idf-simplified-aba19d5f5530

[44] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[45] R. Santos, E. C. Groen, and K. Villela, "An overview of user feedback classification approaches." in *REFSQ Workshops*, 2019.

[46] A. Sharma, Y. Tian, and D. Lo, "Nirmal: Automatic identification of software relevant tweets leveraging language model," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2015, pp. 449–458.

[47] B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas, "Short text classification in twitter to improve information filtering," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 2010, pp. 841–842.

[48] R. E. Stake, *The art of case study research*. sage, 1995.

[49] C. Stanik, M. Haering, and W. Maalej, "Classifying multilingual user feedback using traditional machine learning and deep learning," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2019, pp. 220–226.

[50] A. Syal, "Exploring spacy: Your one-stop library to build advanced nlp products." [Online]. Available: https://towardsdatascience.com/exploring-spacy-your-one-stop-library-to-build-advanced-nlp-products-d242d8d753af

[51] S. Taj, Q. Arain, I. Memon, and A. Zubedi, "To apply data mining for classification of crowd sourced software requirements," in *Proceedings of the 2019 8th International Conference on Software and Information Engineering*, 2019, pp. 42–46.

[52] P. V. Torres-Carrión, C. S. González-González, S. Aciar, and G. Rodríguez-Morales, "Methodology for systematic literature review applied to engineering and education," in *2018 IEEE Global engineering education conference (EDUCON)*. IEEE, 2018, pp. 1364–1373.

[53] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Release planning of mobile apps based on user reviews," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, 2016, pp. 14–24.

[54] P. M. Vu, T. T. Nguyen, H. V. Pham, and T. T. Nguyen, "Mining user opinions in mobile app reviews: A keyword-based approach," *arXiv preprint arXiv:1505.04657*, 2015.

[55] Wikipedia, "Confusion matrix." [Online]. Available: https://en.wikipedia.org/wiki/Confusion_matrix

[56] G. Williams and A. Mahmoud, "Mining twitter feeds for software user requirements," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 1–10.

[57] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 2014, pp. 1–10.

[58] H. Yang and P. Liang, "Identification and classification of requirements from app user reviews." in *SEKE*, 2015, pp. 7–12.

# Appendix A

# Supplemental Information


Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden