GPT Transformer Based Story Points Effort Estimation in Agile Software Development

By

Amna Shahid Cheema

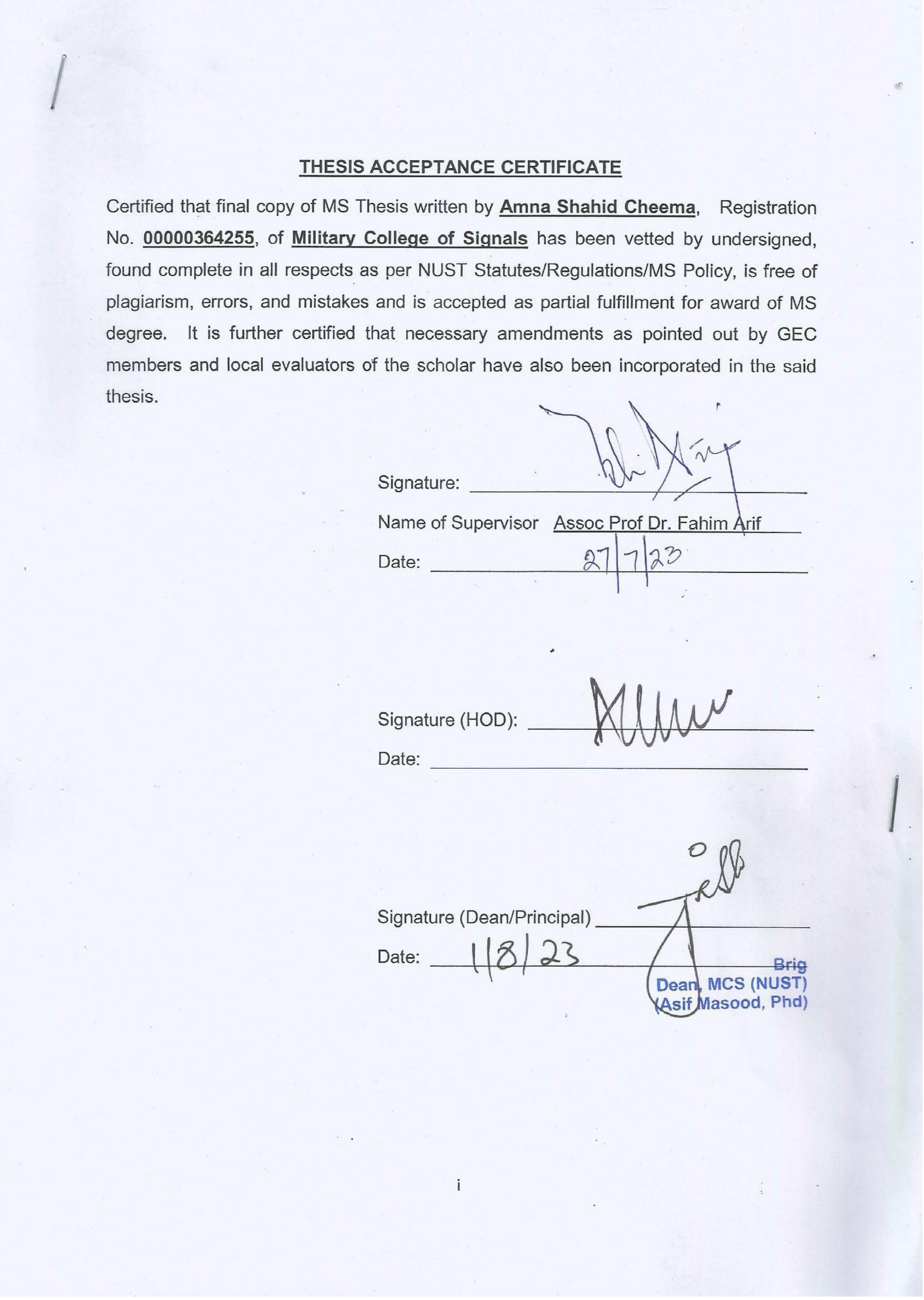
00000364255

Supervisor

Assoc Prof Dr. Fahim Arif

A thesis submitted in the department of Computer Software Engineering, Military College of Signals, National University of Science and Technology, Islamabad, Pakistan for the partial fulfillment of the requirement for the degree of Master of Science in Software Engineering

July 2023



THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS Thesis written by **Amna Shahid Cheema**, Registration No. **00000364255**, of **Military College of Signals** has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations/MS Policy, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members and local evaluators of the scholar have also been incorporated in the said thesis.

Signature: ____________________

Name of Supervisor Assoc Prof Dr. Fahim Arif

Date: 27/7/23

Signature (HOD): ____________________

Date: ____________________

Signature (Dean/Principal) ____________________

Date: 1/8/23

Brig
Dean, MCS (NUST)
(Asif Masood, Phd)

Dedication

This Thesis is dedicated to my beloved Parents, Siblings, Teachers, and Friends who all have been my endless source of love, encouragement, and strength. Your unwavering beliefs in my abilities, countless sacrifices, and relentless support have been the foundation upon which I built my academic pursuits. Without their love and support this research work would not have been made possible.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at Department of Software Engineering at Military College of Signals (MCS) or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at Military College of Signals (MCS) or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Amna Shahid Cheema**

Signature:

Acknowledgments

In the name of Allah (S.W.A), the Creator and Sustainer of the Universe, to whom belongs all glory and power. He alone has the authority to elevate and humble individuals as He pleases. Truly, nothing can be accomplished without His will. From the moment I stepped foot into NUST until the day of my departure, it was by His divine blessings and guidance that I was able to navigate the path of success. His unwavering support and the opportunities He bestowed upon me were instrumental in completing my research journey.

I humbly acknowledge that no words or actions can fully express my gratitude for the countless blessings He has showered upon me throughout this research period. I am indebted to His boundless bounties and am forever grateful for His divine intervention in my academic pursuits. To Allah (S.W.A), I dedicate this thesis as a humble tribute, recognizing His infinite wisdom and benevolence. It is through His mercy that I have reached this milestone, and I pray that my work may be of benefit to others and serve as a means of pleasing Him.

I would also like to express my heartfelt appreciation to my thesis supervisor, **Brig ® Assoc Prof Dr. Fahim Arif**, for his unwavering support and guidance throughout my thesis. His knowledge, expertise, and dedication to his field have been a source of inspiration to me, and I am grateful for the time and effort he invested in my success. Whenever I encountered any difficulties, he was always available to offer his assistance and provide me with insightful feedback.

In addition, I extend my gratitude to my GEC member, **Asst Prof Qazi Mazhar ul Haq** and **Asst Prof Mohammad Sohail**, for their continuous availability for assistance and support throughout my degree, both in coursework and thesis. His expertise and knowledge have been invaluable to me, and I am grateful for his unwavering support and guidance.

Lastly, all praises and thanks be to Allah (S.W.A), the Most Merciful and the Most Gracious.

Amna Shahid Cheema

Implication of Research

The research on GPT-based story points effort estimation in agile software development has profound implications for the industry. By leveraging natural language processing and machine learning, these models enhance accuracy and reduce subjectivity in effort estimation. They provide a standardized, objective approach that saves time and scales well for complex projects. Moreover, GPT-based models continuously learn and improve, integrating seamlessly with existing agile tools. This research opens doors to more efficient project planning, resource allocation, and decision-making, ultimately enhancing the overall effectiveness of the agile development process.

Abstract

The process of accurately estimating the effort required to complete user stories is a crucial activity in software development. It has the potential to significantly impact both the predictability and efficiency of the agile software development cycle. Nevertheless, a considerable proportion of teams are currently facing difficulties in this area, resulting in postponed timelines and unmet deadlines across various domains. Despite the availability of several methodologies aimed at estimating the workload necessary for completing a story point, research has demonstrated that these algorithms are incapable of comprehending the precise contextual requirements of the user. Furthermore, a critical concern pertaining to the techniques of machine learning and deep learning employed in this domain is their elevated time complexity and suboptimal precision. The development of pre-trained transformers, such as GPT, has made a noteworthy contribution to effectively surmounting these challenges. It is possible that certain attention heads may not be effectively contributing to the task of estimating story points, leading to suboptimal outcomes. Notwithstanding the satisfactory performance of several iterations of GPT in previous instances. Through extensive evaluation on 23,313 issues across 16 open-source software projects. The evaluation compared five existing baseline approaches for within- and cross-project scenarios. The results revealed that the GPT2++ approach achieved an impressive accuracy of 92% and an MAE (Mean Absolute Error) of 1.18. Specifically, the GPT2++ approach outperformed existing baseline approaches in two ways: 1. For within-project estimations, the GPT2++ approach was found to be 23% to 59% more accurate than the existing baseline approaches. 2. For cross-project estimations, the GPT2++ approach demonstrated a higher accuracy of 3% to 46% compared to the existing baseline approaches. The ablation study reveals that the GPT-2 architecture employed in this approach significantly enhances GPT2++ by 6% to 47% in terms of performance and boosts the F1 score by 87%. This underscores the remarkable progress of AI in Agile story point estimation.

Table of Contents

Chapter 1… 1

Introduction 1

1.1 Overview.. 6

1.2 Motivation.. 8

1.3 Advantages... 9

1.4 Application... 9

1.5 Problem Statement 10

1.6 Research Objectives.. 10

1.7 Proposed Work..11

1.8 Thesis Organization 11

1.9 Summary… 14

Chapter 2 15

Literature Review… 15

2.1 Overview… 16

2.2 Related Work 16

2.3 Estimation Strategies...16

2.4 Traditional Approaches 18

2.5 Deep Learning Techniques to Estimate Effort 20

2.6 Deep-SE Utilization 22

2.7 Targeted Dataset by Deep-SE and GPT 25

2.8 Hybrid Methodologies 26

2.9 Transfer Learning Techniques 26

2.10 Demerits of Methodologies and Proposed Solutions 27

2.11 Performance indicator 29

2.12 Summary… 34

Chapter 3 35

Proposed Methodology 35

3.1 Overview... 36

3.2 Research Methodology..38

3.3 Proposed Method 38

3.4 Problem Analysis 40

3.5 Text Pre-Processing… 40

3.5.1 Text Cleaning.. 40

3.5.2 Tokenization 42

3.5.3 Optimal Choice of Tokenization 44

3.5.4 Encoding 45

3.6 GPT-2++ Model Integration.. 47

3.6.1 Fine Tuning of GPT2++ Model on Story Points 47

3. 7 Identifying Inefficient Attention Heads..48

3.8 Removing Inefficient Attention Heads... 50

3.9 Model Evaluation 56

3.10 Validation and Analysis.. 58

3.11 Assessment and Analysis...58

3.12 Comparison 58

3.13 Summary… 59

Chapter4 60

Analysis and Results 60

4.1 Overview 61

4.2 Evaluation and Analysis 61

4.2.1 Performance Based on Accuracy and Loss 61

4.2.2 Performance Measure Based on Mean Absolute Error..66

4.2.2.1 With-in Project Assessment 69

4.2.2.2 Cross Project Assessment 70

4.2.3 Performance Based on Sub word Tokenization...71

4.2.4 Performance Based on F1-Score.. 72

4.2.5 GPT2++ Agile Story Point Estimator Tool 76

4.2.5.1 Example Usage of Tool GPT2++.. 78

4.3 Summary…... 79

Chapter 5… 80

Conclusion and Future Work 80

5.1 Overview of Research 81

5.2 Summary of Research Contributions..82

5.3 Conclusion of Research... 84

5.4 Future Work 85

5.5 Summary… 87

References.. 88

List of Figures

Figure 1.1 Working flow of Agile Software Development 16
Figure 1.2 Working Flow of GPT transformer 17
Figure 1.3 Story Points Effort Estimation Process in Agile Software Development 18
Figure 1.4 Multi-head Self Attention in Transformer 19
Figure 1.5 Thesis Organization 27
Figure 2.1 Learning Curves with Lines at MMRE=0.62 was a MMRE of 0.9… 42
Figure 3.1 Flow of GPT-2++ for story point effort estimation by removing inefficient attention heads 43
Figure 3.2 Text cleaning pipeline 43
Figure 3.3 Tokenization of words and paragraphs 44
Figure 3.4 Sub-word model in GPT 45
Figure 3.5 Encoding process in GPT 46
Figure 3.6 Self attention head and Multiheaded self-attention in GPT2++ 47
Figure 3.7 Proposed framework of the improved GPT-2++.......... 50
Figure 3.8 Algorithm of Story Point Effort Estimation using Improved GPT-2++ 55
Figure 4.1 Performance metric Accuracy graph comparison.......... 69
Figure 4.2 Performance metric Accuracy graph w.r.t 20 epochs.......... 69
Figure 4.3 Performance metric Accuracy graph w.r.t 30 epochs.......... 70
Figure 4.4 Performance metric Accuracy graph w.r.t 40 epochs.......... 71
Figure 4.5 Performance matric Mean Absolute Error comparison… 75
Figure 4.6 Shows the relative improvement in MAE for within-project estimation 77
Figure 4.7 MAE for cross-project estimate using our GPT2++, Deep-SE, and ABE0 78
Figure 4.8 shows the effect on our GPT2++ model when we alter either the tokenization 80
Figure 4.9 Performance metric F1 Score graph comparison… 83

List of Tables

Table 2.1 Detailed Summery of Researchers Findings and Limitations 48

Table 4.1 Performance metric Accuracy table comparison...68

Table 4.2 Performance metric Accuracy table comparison w.r.t 20,30,40 epochs.. 71

Table 4.3 Performance metric Mean Absolute Error table comparison.. 74

Table 4.4 Performance metric F1 Score table comparison..82

Table 4.5 snapshot of the proposed GPT2++ tool...83

List of Abbreviation

GPT	Generative Pre-trained Transformer
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
DSE	Deep-Semantic Extraction
SE	Software Engineering
RHWN	Recurrent Highway Network
SVM	Support Vector Machine
GloVe	Global Vectors for Word Representation
HAN	Hierarchical Attention Network
ASD	Autism spectrum disorder
NN	Neural Network
MMRE	Mean Magnitude of Relative Error
SP	Story Point
ML	Machine Learning
TF-IDF	Term Frequency-Inverse Document Frequency
NB	Naive Bayes,
DT	Decision Trees
K-NN	K-Nearest Neighbors Algorithm
MAE	Mean Absolute Error
ABM	Algorithm-Based Model
FP	Functional Point
CP	Case Point

LS	Large Scale
SDLC	Software Development Life Cycle
TTF	Total Technical Factors
TUCP	Total number of Use Case points
TEF	Total Environmental Factors
TEC	Total Estimated Cost
WBS	Work Break Down Structure
AEE	Analogy Effort Estimation
REM	Regression-Based Estimation Model
SSEM	Software Sized based Estimation Model
FEM	Functional Estimation Model
SPE	Story Point Estimation
ANOVA	Analysis Variance
RF	Radio Frequency
ATLM	Automated Testing Lifecycle Management
LR	Long Range
RF	Random Forest
ATLM	Automatically Transform Linear Models
ABE	Analogy-Based Estimation
NMT	Need More Time

Chapter 1

Introduction

Chapter 1

Introduction

1.1 Overview

The accurate prediction of the complexity of user stories and the corresponding effort required for their completion is crucial for the successful execution and predictability of the software development cycle (figure 1.1). Notwithstanding this fact, a considerable proportion of enterprises still encounter challenges in this domain, resulting in time lags and failure to meet established timelines. Regrettably, the algorithms currently at our disposal for evaluating the requisite labor involved in completing story points encounter difficulties in comprehensively discerning the user's contextual expectations, thereby yielding suboptimal outcomes.

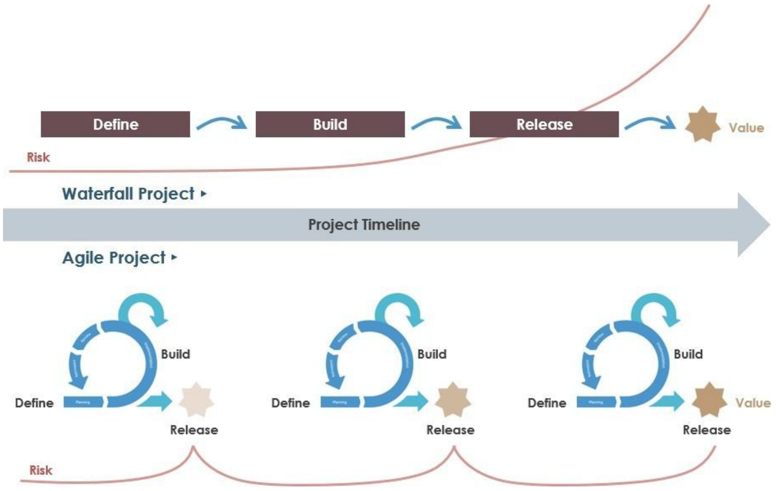


Figure 1.1 Working of Agile Software Development.

Furthermore, research has demonstrated that the utilization of machine learning and deep learning algorithms for estimation purposes may present challenges due to their significant time complexity and limited accuracy. Pre-trained transformer models, specifically the Generative Pre-trained Transformer (GPT) described in figure 1.2, have demonstrated their ability to address the challenges associated with this scenario. GPT models exhibit promising potential in enhancing the precision of story points estimation and have demonstrated remarkable efficacy in diverse natural language processing domains.

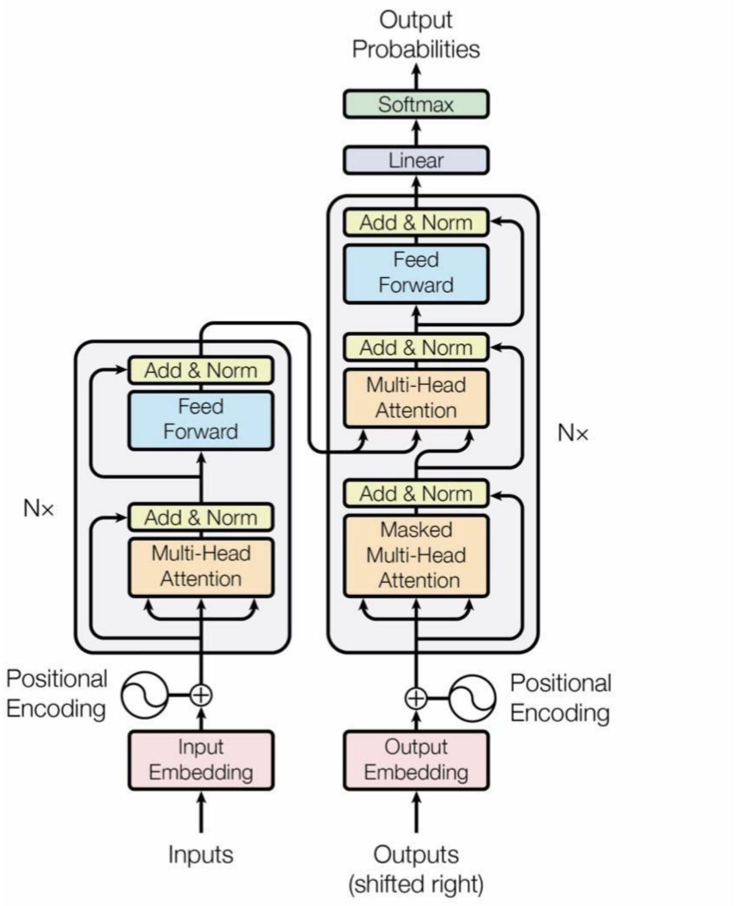


Figure 1.2 Working Flow of GPT Transformer.

The application of said models presents the prospect of enhancing the evaluation of various other attributes of natural language. In order to optimize the performance of these models, it is imperative to identify and eliminate any attention heads that are not effectively fulfilling their intended purpose.In the industry of software project management, precise estimation of the workload is crucial for effective planning and monitoring. Cost and time overrun issues have historically been a problem for software development projects. According to the results of a study by McKinsey and the University of Oxford [1], notable software projects typically overspend their budgets by 66% and their schedules by 33%. As per the results of an independent study that examined 1,471 software projects and yielded similar outcomes, roughly 16.67% of software projects encountered cost overruns of 200% and time overruns exceeding 70%. [2] The study was carried out on an equivalent number of software projects. Effort estimation activities are imperative for the adequate planning and administration of a software project. In order to ensure timely completion of the project within the allocated budget, it is imperative to undertake the following measures: [3] [4] [5]. The assessment of the amount of work required can serve as the basis for a diverse range of activities pertaining to planning, scheduling, budgeting, and costing, as stated in reference [6].

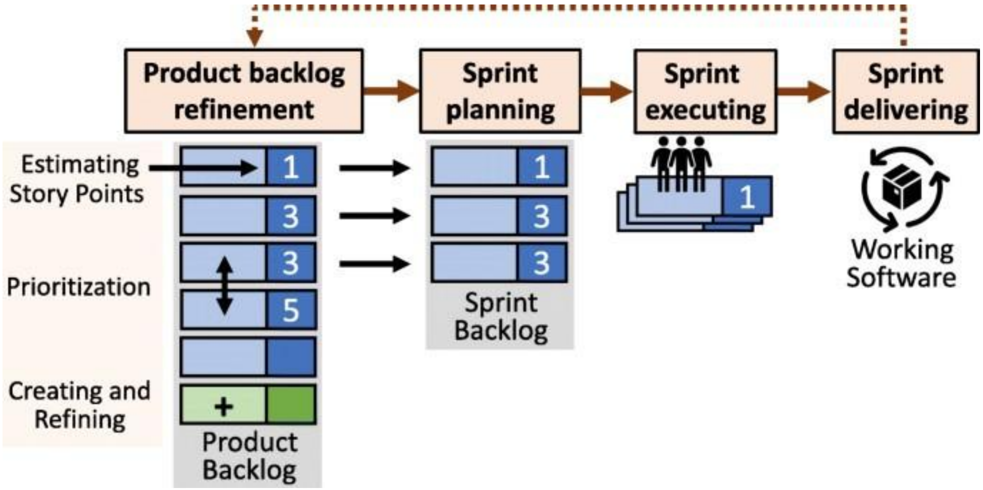


Figure 1.3 Story Points Effort Estimation Process in Agile Software Development

These activities (figure 1.3) have the potential to be executed in diverse settings by a broad spectrum of individuals. Inaccurate estimations possess the capacity to adversely affect the outcomes of the project [7]. This is due to the fact that such estimations can potentially result in detrimental consequences. The objective of this study is to enhance the accuracy of story point estimation through the elimination of ineffective attention heads in the GPT-2++ model.

Ultimately, it is expected that this will result in more precise estimations of the workload required to satisfy storypoints with reduced time requirements compared to previous methods. Our research in the domain of story-point estimation surpasses prior efforts by incorporating pre-trained transformers, such as GPT-2++, to circumvent the limitations imposed by previously developed techniques. This is undertaken as a means of surmounting the limitations imposed by pre-existing methodologies.This inquiry is primarily focused on achieving one of two primary objectives. One of our key goals when it comes to estimating how long it will take to finish a specific user story is to have a better knowledge of the challenges that are faced by software development teams. This will allow us to predict more accurately the estimated time required. Looking at the difficulties that arise when machine learning and deep learning are put into practice, as well as the limits of the approaches that are now accessible. Finally, a method is introduced for increasing the accuracy of the estimation of story points that is produced by the GPT-2++ model by getting rid of the attention heads that aren't very efficient.Explanation of attention described in figure 1.4.

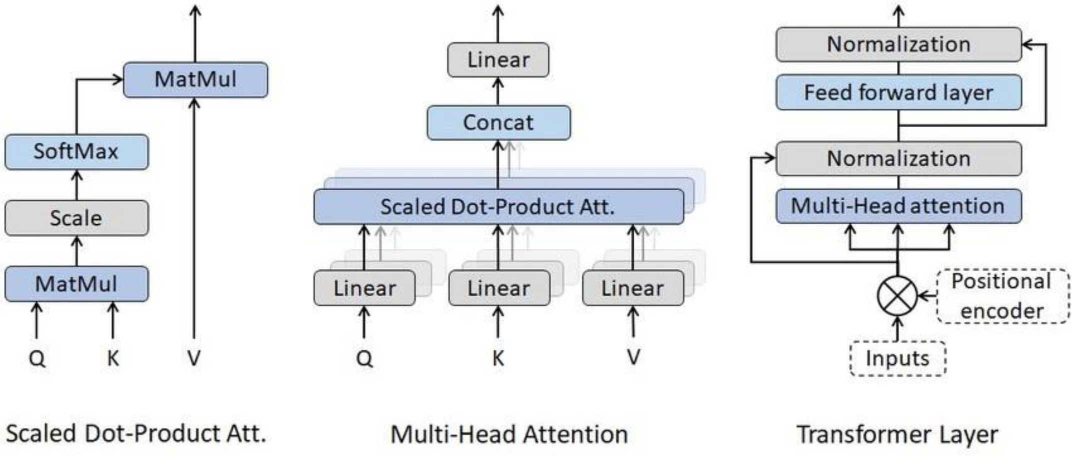


Figure 1.4 Multi-head Self Attention in Transformer GPT-2

This technique should provide significant benefits to the estimation of story points, both in terms of accuracy and efficiency; therefore, project management should become more reliable and resources should be used more efficiently. Teams that are now working on the production of software in the "real world" may be able to put the results of this study to good use. An accurate estimation of story points makes it feasible for teams to more effectively plan and monitor progress, which in turn leads to more efficient project execution and fewer delays. In addition, the study provides a contribution to the larger field of software estimation by investigating new approaches for strengthening attention processes in pre-trained transformers. This was done to make the work more accurate. The subsequent sections will provide an overview of the challenges associated with story-point estimation, review prior studies on machine learning methods and pre-trained transformers, and present our approach to enhancing story-point estimation via the identification and elimination of unnecessary attention heads. Ultimately, presented a succinct summary of the results and explore potential avenues for future investigation in this domain.

1.2 Motivation

The selection of the topic "GPT Transformer Based Story Points Effort Estimation in Agile software Development" is justified for several reasons, one of which is that it will serve as the central theme of this research:

- Precisely determining the quantity of story points necessary for a software development undertaking is a crucial measure; however, the process of estimation is widely recognized as a difficult task and frequently results in unmet deadlines and additional delays.
- GPT-2++ is a state-of-the-art language model that has demonstrated remarkable potential in various natural language processing domains. One of the applications that is under consideration pertains to the estimation of story points, which has demonstrated remarkable potential across all domains.
- It is probable that some attention heads in GPT-2++ are not efficiently contributing to the computation of story points, leading to suboptimal results. Assuming this hypothetical scenario, the outcomes would ensue.

- The removal of these inefficient attention heads could potentially improve the performance of GPT-2++ in terms of story point estimation. Consequently, this would result in estimations that are more accurate and efficacious.
- The potential implications of the findings of this research could have significant consequences for the reliability and efficacy of software development in the future.
- The domains of artificial intelligence and natural language processing have been increasingly attracting the interest of software developers. The novelty of this concept notwithstanding, it has received limited scholarly attention in the past, rendering it a compelling subject for academic inquiry.

1.3 Advantages

The implementation of our revised GPT-2++ recommendation when estimating the quantity of story points to be accomplished yields various advantages:

- **Improved Accuracy:** One potential approach to enhancing the performance of GPT-2++ in the context of story point estimation involves eliminating attention heads that do not significantly contribute to the model's accuracy. As a result of this, we will have an improved ability to predict pivotal moments within the story.
- **Enhanced Predictability:** Enhancing the predictability and productivity of software development can be achieved through the provision of estimates that are both precise and efficient in evaluating the immense amount of work involved. necessary to finish user stories.
- **Reduced Time and Resources:** Making the switch to the enhanced GPT-2++ will reduce the amount of time and effort needed to train a model from the very beginning of the process.
- **Ease of Use:** The approach that is going to be proposed for the purpose of story point estimation will be user-friendly, and it will be able to be tweaked so that it will work more effectively on certain datasets.

- **Generalization Ability:** The model that have suggested is adaptable and can easily generalize to new sets of information, both of which make it an outstanding candidate for usage in a wide range of natural language processing applications.

- **Better Decision Making:** Organizations can enhance their decision-making regarding software development by utilizing precise and effective estimates of the workload required to complete a user story.

- **Improved Competitiveness:** Businesses can improve their global competitiveness if they create software that is not only more trustworthy but also more efficient.

1.4 Application

The proposed model, referred to as the extended GPT-2++, has been designed to estimate the effort required to complete a story point. This model has several notable applications in the field of software development.

- **Agile Software Development:** Various methodologies have been developed for facilitating the development of agile software, including Scrum, Kanban, and Lean. These approaches may potentially derive advantages from the implementation of this strategy.

- **Project Management:** The adoption of this model by project managers can enable the estimation of the duration necessary to complete specific user stories, thereby facilitating more accurate allocation of resources and time.

- **Quality Assurance:** Professionals specializing in quality assurance can utilize this framework to furnish a projected expense for the examination and authentication associated with user stories, thereby enabling more accurate preparation and administration of stated projects.

- **Software Testing:** Software testers may use this strategy to make an educated guess as to how long it will take to test each individual user story. This gives them the ability to manage resources and activities in a more effective manner and better prioritize tasks.

- **Software Maintenance:** By providing estimates of the amount of time and resources needed to solve certain bugs and other problems, this method may be used to enhance the planning and management of software maintenance activities. This can be done by improving planning and management of software maintenance tasks.
- **Business Analysis:** The model has the potential to be utilized in the context of business analysis, with the aim of providing an approximation of the workload associated with the execution of specific business demands. Additionally, it can aid in the optimization of planning and management strategies for the implementation of business requirements. Furthermore, it can be utilized to furnish a projection of the duration necessary to accomplish the undertaking.

1.5 Problem Statement

The present challenges in accurately estimating the requisite level of complexity and workload to fulfill user stories are resulting in software development teams falling behind schedule and failing to meet deadlines. This presents a challenge, as there are difficulties in accurately predicting the degree of complexity and effort necessary to fulfill user stories. There exists a possibility that certain attention heads may not be effectively contributing to the task of estimating story points, leading to suboptimal outcomes. Regardless of the fact that diverse iterations of GPT have exhibited commendable performance in previous studies, By eliminating non-constructive attention heads in the GPT-2++ model, it is possible to enhance the precision of story point estimation as well as the predictability and productivity of the development cycle. Enhancing the precision of the story point estimation procedure is an objective to be pursued.

1.6 Research Objectives

The following might be the main objectives of the study on the use of modified GPT-2++ for assessing story point effort in software development by removing superfluous attention heads:

- This study aims to investigate if removing the ineffective attention heads from GPT-2++ may enhance its performance in calculating story points .

- To compare the performance of the GPT-2++ model with increased attention heads to the standard GPT-2++ model and to established techniques for story point estimation.
- The aim is to identify the GPT-2++ model's most effective and significant attention heads in terms of their contribution to the estimate of story points.
- The goal of this study is to ascertain if using a more advanced version of GPT-2++ to generate story point estimation to improve the predictability and effectiveness of the software development process.
- With the intention of improving software development, the goal is to propose novel approaches for anticipating story points using the upgraded GPT-2++ model.
- This research's goal is to examine and evaluate any potential negative effects of using improved GPT-2++ for estimating story points. This research also tries to suggest viable answers to these problems. The goal is to find, investigate, and evaluate any possible benefits of using upgraded GPT2++ for story point estimation.

1.7 Proposed Work

The present study introduces a novel approach aimed at enhancing the precision and efficiency of effort estimation in software development. This approach leverages the capabilities of preexisting transformers, specifically GPT-2++, to achieve its objective. The limitations of current estimation algorithms and the restrictions imposed on this field by machine learning and deep learning methodologies are acknowledged. It is hypothesized that the utilization of the sophisticated natural language processing functionalities provided by GPT-2++ will enable a more accurate interpretation and contextualization of customer requirements, thereby facilitating the provision of more precise estimates. The methodology employed involves data cleansing, integration of GPT-2++ into the estimation pipeline, model fine-tuning, and performance evaluation. The implementation of this approach enables project planners and managers to generate more accurate predictions regarding the quantity of story points required for a given task, and subsequently allocate resources in accordance with these estimations.

1.8 Thesis Organization

This research paper's subsequent sections provide a thorough summation of supporting evidence for the main premise. The sections outlined below are depicted in Figure 1.5.

Chapter 1:

The significance of making an accurate effort estimation in agile software development. Difficulties that companies have in accurately calculating the amount of time and effort required for user stories. Limitations of currently available algorithms and methods to machine learning. Pre-trained transformers, GPT models, have the potential to improve estimation. The purpose of this research is to improve story points estimation by getting rid of attention heads in GPT2++ that aren't very effective. And also, the significance of precise estimating in relation to the planning, monitoring, and carrying out of projects.

Chapter 2:

A comprehensive review of the difficulties associated with software project management, including budget and time overruns. Previous research on software estimating and the consequences of using estimates that are too wrong. Analysis of current methods for estimating story points, including those based on machine learning and deep learning. An overview of pretrained transformers and the positive impact they have had in various natural language processing applications.

Chapter 3:

An explanation of the study approach that was used to accomplish the outlined objectives. Brief description of the procedures of collecting data, doing preprocessing, and training models in the GPT-2++ paradigm, the identification and assessment of attention heads. Also, the actions performed to eliminate attention-grabbing phrases that aren't working and to improve assessment of story points.

Chapter 4:

Presentation of the results from the research, together with an analysis of those findings. Detailed Comparison of the newly developed method for estimating story points with the methods used in the past. Analysis of the precision and effectiveness that may be attained by the removal of attention head clutter. Discussion of the significance of the study results as well as possible applications of the findings.

Chapter 5:

A synopsis of the study's goals, its procedures, and its most important results. The significance of providing precise estimation of effort for software development projects. A contribution to the area of software estimation as well as the implementation of pre-trained transformers. Suggestions for doing further study and adventuring in this domain. The research presented in this thesis follows a natural progression from the introductory and background sections to the literature review, the research objectives and methodology, the presentation and discussion of the results, and finally the conclusion and suggestions for future study.

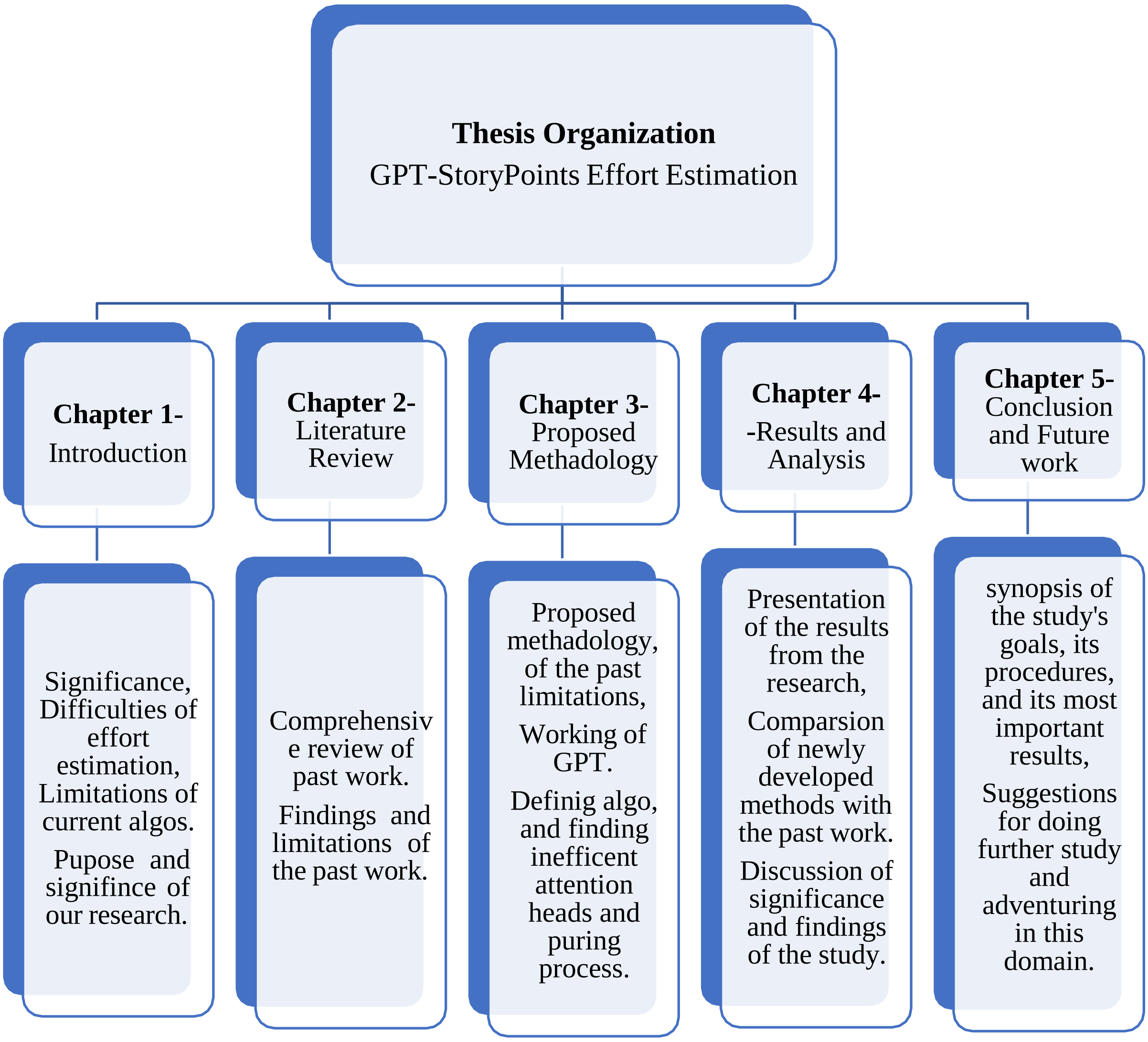

Thesis Organization
GPT-StoryPoints Effort Estimation
Chapter 1- Introduction
Chapter 2- Literature Review
Chapter 3- Proposed Methadology
Chapter 4- -Results and Analysis
Chapter 5- Conclusion and Future work
Significance, Difficulties of effort estimation, Limitations of current algos.
Pupose and significe of our research.
Comprehensive review of past work.
Findings and limitations of the past work.
Proposed methadology, of the past limitations,
Working of GPT.
Definig algo, and finding inefficent attention heads and puring process.
Presentation of the results from the research,
Comparsion of newly developed methods with the past work.
Discussion of significance and findings of the study.
synopsis of the study's goals, its procedures, and its most important results,
Suggestions for doing further study and adventuring in this domain.

Figure 1.5 Thesis Organization

1.9 Summary

Chapter one presents a comprehensive analysis of the difficulties that businesses encounter when evaluating the complexity and time required for the effort estimation. Additionally, it highlights the importance of accurately assessing story points during the software development process. The declaration highlights the limitations of current machine learning algorithms and methodologies, while showcasing the capabilities of pre-trained transformers, specifically GPT models, in enhancing estimations. The present chapter explicates the aim of the investigation, which is to augment the story points effort estimation through the elimination of attention heads in GPT-2++ that exhibit low efficacy. Furthermore, it underscores the significance of precise estimation in the domains of project planning, monitoring, and execution. The chapter that follows discusses the research objective, which pertains to enhancing story point estimation through the elimination of underperforming attention heads in GPT-2++.

Chapter 2

Literature Review

Chapter 2

Literature Review

2.1 Overview

This chapter will discuss various challenges that agile software development currently encounters in relation to estimating story points. It will encompass an exploration of diverse methodologies proposed by scholars, along with a comprehensive evaluation of their advantages, disadvantages, and in-depth analytical examination.

2.2 Related Work

The estimation methodologies were broadly categorized into three distinct groups: expert-based techniques, model-based techniques, and hybrid techniques. The predominant approach utilized in practical applications was commonly referred to as expert-based processes, also known as methodologies, that relied on human expertise to formulate estimations [8] [9]. However, developing an assessment that relied on the viewpoints of professionals could potentially require a significant amount of time and financial resources. Moreover, achieving success necessitated continuous access to essential experts. Diverse model-based approaches exhibited varying degrees of adaptability in incorporating user input and leveraging prior models, yet they uniformly relied on insights gleaned from previously completed endeavors.

2.3 Estimation Strategies

It was noted that companies had been adopting Agile for over 20 years, experiencing different trends and evolutions. Scrum emerged as a popular framework within Agile, encompassing iterative development, focused work aim, cooperation, client participation, face-to-face communication, minimal documentation, frequent testing, collective responsibility, and knowledge transfer [10]. A comparison was made between the Traditional and Agile planning procedures, analyzing algorithm-based estimating techniques like Source Line of Code (SLOC), Functional Point, Object Point, and Constructive Cost Model (COCOMO), as well as non-algorithmic techniques such as Expert Judgment and the Analogy approach Wideband Delphi [11].

Estimating mobile development estimates involved utilizing the COSMIC (Common Software Measurement International Consortium) Functional Point Measurement (FMS) methods. Specific Function, Data Manipulation Function, Inquiry Function, User Support Function, and Developed View Functionality were among the functional process measurements used in Agile estimations [12]. The study further compared estimation models like Use Case Point (CP), Functional Point (FP), COCOMO, Algorithm-Based Model (ABM), Expert Judgment Model, and Estimation [13]. The Agile approach scored higher on flexibility and collaboration when compared to Traditional methods. The potential for failure in Agile projects was evaluated based on factors such as team meetings, site visits, training, documentation, communications clarity, and project personnel [14].

To achieve better realization at a low cost, a hybrid approach recommended the use of development methodologies with features like Large Scale (LS), High Reliability, High Productivity, High Estimation Accuracy, Early Realization, and Ease of Change [15]. Component estimate systems were developed for multi-agent systems, and ontologies and other knowledge-based approaches were found to improve work estimates in Agile development [16]. The use of expert-based estimations was shown to increase accuracy in Agile project time estimates [17]. A reference model for estimates was created based on an analysis of narrative point life cycle and frequently used story points [18]. Additionally, a comparison was made between the benefits and drawbacks of different SDLC methodologies, including Waterfall and Agile [19].

An algorithm for estimating costs, called the Algorithm for Estimating Costs (TEC), aimed for precision by considering environmental and technological factors. Total Technical Factors (TTF), Use Case Points (TUCP), and Environmental Factors (TEF) formed the foundation of this algorithm [20]. Weighted and Complexity Factors were employed to determine Functional Points for various needs, injecting infusion factors to the Functional Point projections [21]. It was found that Agile projects had a higher success rate compared to Waterfall, with a factor of two difference in success rates according to the Standish Group's 2018 Chaos Report [22]. Various estimation models, including analogy-based effort estimation (AEE), regression-based estimation model (REM), software-sized estimation model (SSEM), functional estimation model (FEM), work breakdown structure (WBS), and story point estimation (SPE), were discussed, with WBS and SPE being recommended as the best fit for Agile projects [23]. Furthermore, it was discovered that using the average size of User Stories instead of the Consensus size led to reduced overall accuracy [24]. Research on Agile mobile app development indicated that Poker was used in 63% of projects, Analogy in 47%, and expert judgment in 38% [25].

2.4 Traditional Approaches

The COCOMO model for construction cost, which was developed by Boehm and colleagues in 1998, exemplified a fixed model due to its utilization of fixed parameters and their interactions. Over time, this particular model gained significant popularity. The development of these estimation models involved utilizing data gathered from a diverse array of previously executed projects. Consequently, their relevance was often limited to projects that shared similarities with the one utilized for constructing the model, thereby restricting their utility. Various methodologies were proposed in the literature to address this problem, including regressionbased approaches [26, 27], neural network models [28, 29], fuzzy logic Bayesian belief networks [30], and analogical reasoning techniques.

It was important to note that a universal approach might not have been optimal for all types of projects [31]. Several recent studies proposed the integration of results from multiple estimators, similar to the concepts presented in studies utilizing hybrid techniques [32, 33].

Considerable efforts had been devoted to project estimation in general; however, only a negligible fraction of these endeavors had been directed towards developing models that were tailored to the requirements of agile project management. This modification was deemed necessary based on the findings of reference [34], which indicated that contemporary business enterprises necessitated the adoption of alternative methodologies for cost estimation and planning. Advanced techniques utilized algorithms for machine learning to offer support for agile project work estimations. A recent publication [35] discussed a certain strategy for facilitating the construction of a story-point estimate model. This strategy entailed the derivation of TF-IDF features from the problem statement to achieve the intended outcome. Subsequently, univariate feature selection methodologies were employed to incorporate the recently generated features into classifiers, such as the support vector machine (SVM). In another study [36], the CFP method was utilized to precisely gauge the duration required to complete an agile project. Regression models and neural networks were employed within an iterative software development framework to construct a predictive model for estimating the amount of labor needed [37]. This model provided estimations for the aggregate quantity of human labor that would be required. Unlike the conventional approach of creating an estimation model at the end of a project, the iterative approach involved constructing such a model subsequent to each iteration with the aim of projecting the requisite workload for the forthcoming iteration, facilitating the development of more precise project planning.

The Bayesian network model, as introduced in reference [38], held utility for software development endeavors that employed the Extreme Programming methodology within the domain of agile software development. However, the utilization of a multifaceted approach in their model, encompassing factors such as process efficacy and potential for future expansion, necessitated the acquisition of novel data and significant alterations. In another examination conducted by researchers [39], Bayesian networks were employed to examine the interrelationships present within a software development project based on Scrum methodology. The primary aim of a software development project that employed the Scrum methodology was to identify potential issues at the earliest possible stage to avert more significant setbacks. The authors conducted a simulation to examine the impact of product quality on the advancement of sprints and the quality of sprint planning.

The methodology presented in this context differed from its predecessors in two notable aspects. Firstly, deep learning techniques were employed to automatically obtain semantic characteristics that delineated the actual connotation of problem descriptions. Secondly, these characteristics were utilized to estimate story points, thereby distinguishing this approach from alternative methodologies. The successful implementation of this method required the completion of both steps. The matter of ascertaining the appropriate amount of time and/or risk that should be allocated towards rectifying a fault had previously been the focus of scholarly inquiry and investigation [40-43].

2.5 Deep Learning Techniques to Estimate Effort

In contrast, the agile software development methodology advocated the utilization of "story points" as a means of estimating work. The implementation of LSTM had been observed in various language models [44], voice recognition systems [45], and video analysis tools [46], among other applications, indicating its potential benefits in diverse domains. The Deep-SE model exhibited a high degree of generalizability, as it could be applied to a diverse array of tasks by mapping textual data to either a numerical score or a categorical label. The attribute of generalizability endowed the system with greater adaptability and utility. Examples of academic tasks where this attribute could be applied included assessing written compositions and conducting sentiment analysis.

Deep learning had been increasingly adopted in the field of software engineering. In previous work, a universal deep learning framework utilizing LSTMs had been presented to depict the software engineering process [47]. The outcomes of this research were evident in the structure. The simulation of a programming language using recurrent neural networks was conducted by the authors in reference [48]. The RNN models were modified to enhance their ability to detect occurrences of code replication in subsequent research. The team had also developed a linguistic model for code using LSTM technology, and the research outcomes were disseminated in a publication [49]. The purpose of releasing this model was to share the research findings. The LSTM model outperformed the RNN model in terms of accuracy.

In reference [50], the generation of the output sequence was facilitated by a specific Recurrent Neural Network (RNN) Encoder-Decoder model. This model, consisting of an encoder RNN and a decoder RNN, could receive a query in natural language related to the API and generate a series of API calls as output. Reference [51] presented research that utilized a Recurrent Neural Network (RNN) Encoder-Decoder model to address prevalent issues in the realm of C programming. The Deep Belief Network, a renowned deep learning model introduced in 2006, had been effectively integrated into functional software [52].

To summarize, the Agile software development cycle consisted of four main stages: product backlog refinement, sprint planning, sprint execution, and sprint delivery. The collaborative effort between product owners and customer representatives led to the creation of a prioritized list of software needs, known as the product backlog [53]. During the product backlog refinement process, the team conducted a thorough review of the listed tasks and implemented necessary modifications. Work breakdowns required the team to establish a set of activities exhibiting similarities and differences, often referred to as narratives or task components. Thoroughness was particularly emphasized for significant assignments like Epics, which provided a comprehensive account of a feature. Task prioritization and sequencing decisions were influenced by the anticipated workload. The final stage involved organizing sprints, where the team defined the sprint objective, evaluated their capacity, selected work items for the sprint backlog based on capacity, and conducted rapid iterations to produce a fully functional product [54].

2.6 Deep-SE Utilization

One approach that was used in providing an indication of the level of difficulty associated with a given task was to assign a designated quantity of Story Points (SP) to each task. It was observed that the term "issue" was significantly more frequent in usage than the term "task" within JIRA. Teams employed various strategies, including Planning Poker, Analogy, and expert judgment, based on factors such as total work, complexity, risk, and uncertainty, to estimate narrative points, as stated in reference [55].

Usman et al. [56] indicated that subjective assessments, which relied on the expertise of domain experts, could introduce bias. They arrived at this deduction through their academic study. It was recognized that inaccurate estimation of the number of stories needed to complete a sprint could lead to negative consequences such as decreased efficiency, increased costs, project failure, client dissatisfaction, and potential business closure.

A recent study introduced a novel approach known as Deep-SE[57], which utilized comprehensive deep learning methods to forecast the quantity of story points involved in agile projects. The validation of the model was accomplished through the utilization of data derived from both ongoing and completed projects.

2.7 Targeted Dataset by Deep-SE and GPT

The Deep-SE model had a corpus of 23,313 bugs, which were collected from 16 distinct opensource projects using the JIRA bug tracking software. Each issue consisted of a title, a synopsis, and a significant plot advancement, all of which were relevant to the matter at hand. Figure 2.1 depicted the JIRA issue and its corresponding pivotal moment. Deep-SE employed a deep learning framework, specifically the Long Short-Term Memory (LSTM) and Recurrent

Highway Network (RHWN), to capture semantic features representing the meaning of a given problem. The issue was duly reported, and Deep-SE was requested to investigate.

Deep-SE demonstrated superior performance compared to other machine learning-based techniques, such as LSTM+RF, BoW+RF, Doc2Vec+RF, and TFIDF+SVM [58], with an average Mean Absolute Error of 2.08, indicating higher quality outcomes. The Deep-SE process involved four distinct phases. The initial step involved adding written content to the designated area, which was recognized as a challenging task due to the need to comprehend problem statements expressed in simple language.

Deep-SE utilized an unannotated corpus of specialized data, such as issue reports, to acquire knowledge of the distributed representation of words. This was achieved by creating a pretrained language model and investigating lexical representations in a distributed manner. One approach was to integrate the title and description of the report into a unified document, with the description preceding the title. Words in the problem report were represented as vectors with continuous real values.

In the second step, long-short-term memory (LSTM) was employed to generate a representation of the document. Deep-Semantic Extraction utilized LSTM units, a specific type of Recurrent Neural Network (RNN), for textual analysis. The output state vectors of the LSTM were then consolidated into a single vector representing the document.

The third step involved the use of the RHWN method to develop a detailed model of the data. Deep-SE mitigated the issue of overfitting by subjecting the document vector to multiple refinements through a Recurrent Highway Network (RHWN), resulting in a conclusive vector representing the document. This characteristic made Deep-SE resilient to overfitting.

In the fourth step, regressors were utilized. Deep-SE employed a feedforward neural network with a linear activation function to approximate the narrative point based on the document vector.Software development work estimation could be broadly categorized into three approaches: expert-based, model-based, and hybrid approaches. The prevalent methodology involved expert-based approaches that relied on human expertise to provide approximations [59] [60]. Expert-based estimating required the involvement of specialists for accurate projections. Model-based methodologies utilized past data, but the construction of individual models varied. The CO-COMO model, a widely recognized construction cost model [61], was a static model with predetermined components and variables. The predictive models were developed by leveraging data from multiple studies but were typically limited in applicability to the specific projects for which they were initially developed. Scholars employed various methodologies, such as regression [62], neural networks [63, 64], fuzzy logic [65], Bayesian belief networks [66], analogy-based approaches [67], and multi-objective evolutionary methods [68], to construct distinctive models. However, a single tactic could not be universally effective for all project types [69]. Recent research [70] advised aggregating estimates from multiple estimators in a sequential manner.

2.8 Hybrid Methodologies

The present study and its associated literature [71, 75] utilized hybrid methodologies that integrated expert opinions with readily available data. While there was a significant corpus of literature on project estimation in general, less attention was given to formulating models specifically tailored to agile projects. Agile, dynamic, and incremental projects required alternative planning and estimation approaches [76]. Machine learning methodologies were increasingly used for task estimation in agile projects. A recent research publication [77, 78] proposed a method for extracting TF-IDF features from issue descriptions to construct a model for narrative point estimations. The retrieved features underwent standardized selection before being inputted into regression models like the support vector machine.

There was comparatively less attention paid to the development of models specifically tailored to agile projects, despite a significant corpus of literature on project estimation in general. Agile, dynamic, and incremental projects necessitated alternative methodologies for planning and estimation [79]. Machine learning methodologies were employed to facilitate task estimation in agile endeavors. A recent research publication [80] introduced a novel approach for constructing a model for narrative point estimates by extracting TF-IDF features from issue descriptions. The features underwent standardized selection before being inputted into regression models like the support vector machine.

In Extreme Programming software projects, an iterative development methodology, the authors presented a Bayesian network model for effort estimation, as documented in reference [81]. However, their methodology relied on factors such as process efficiency and improvement, requiring thorough experimentation and fine-tuning. Bayesian networks were commonly used in Scrum-based software development projects to model interrelationships and identify potential challenges. For instance, the quality of sprint progress and planning could impact the final product's quality. Recent developments involved the utilization of deep learning techniques to automatically learn semantic features capturing the essence of problem descriptions. This approach made significant progress in estimating issues based on narrative points, surpassing previous efforts. Previous studies explored the estimation of defect resolution time and the potential risks associated with resuming issue resolution after a pause [82, 83].

2.9 Transfer Learning Techniques

The proposed model deviated significantly from [84] by utilizing transfer learning techniques with GloVe and pre-trained embedding vectors to expedite the training process. Word2Vec and GloVe were contemporary methodologies generating superior vector representations [85, 86]. GloVe outperformed Word2Vec due to its comprehensive dataset of term occurrences from various regions, as demonstrated by Pennington et al. [87]. GloVe [88] served two purposes: similarity and entity identification, based on statistical analysis of word-word cooccurrences within a corpus. Empirical evidence showed that the model introduced in reference [89] exhibited superior performance compared to previous ones [90–95]. Notably, this model enabled end-to-end trainability without requiring human feature engineering, starting from raw input data and culminating in prediction outcomes. The proposed model employed deep learning and a hierarchical attention mechanism to detect significant phrases and clauses. The Hierarchical Attention Network (HAN) effectively captured the fundamental principles of document organization. Document structure was hierarchical, with words forming sentences and sentences forming a complete document. To create a document representation, individual sentence representations were generated and aggregated into a cohesive document representation. The density of information conveyed by individual words and phrases varied throughout the text. Constructing a final document vector involved aggregating significant sentence vectors, composed of essential word vectors. Three surveys were retrieved that examined distinct facets of estimation in individuals with Autism Spectrum Disorder (ASD).

2.10 Demerits of Methodologies and Proposed Solutions

The study, as reported in 2005 [96], encompassed feedback from project managers employed at 18 commercial enterprises located in Norway. A comparison of schedule and effort overruns was conducted across 52 projects, utilizing both adaptive (incremental, agile) and sequential (waterfall) process models. Projects employing a flexible process model exhibited a lower incidence of effort overruns compared to those utilizing sequential approaches, according to the study. The findings of this survey were utilized in a scholarly investigation [97] that analyzed various aspects of software estimation within the Norwegian corporate domain, including effort and schedule overruns, estimation techniques, estimation competence, and related factors. However, the present investigation did not distinguish between agile and waterfall methodologies.

The subsequent survey [98] examined the impact of consumer engagement on project prolongation. Comprehensive interviews were conducted with agile project managers employed by a Norwegian medium-sized enterprise, using a data set of 18 projects. The study's findings suggest that collaboration with the client through regular communication can decrease the frequency of effort overruns. Another survey [99] focused on the anticipated duration for executing a user narrative. The findings revealed that the utilization of coarsegrained user stories by developers was associated with a higher incidence of obstacles, including estimation concerns. Despite the widespread use of surveys to evaluate estimation in Autism Spectrum Disorder (ASD), none provided a comprehensive report on estimation methods, predictors, reliability, or the developmental context in which they are employed. To address this, an empirical investigation was conducted involving practitioners from software enterprises utilizing agile methodologies or practices, irrespective of geographical location.

The efficacy of a project plan depends significantly on the utilization of a precise and reliable approach for estimating labor demands. Studies in the field of agile methodology have shown that effort assessment often involves the use of narrative points. Estimation techniques based on human judgment, such as Planning Poker, are commonly employed [100]. Limitations associated with these techniques have been the focus of inquiries. Research findings [101] demonstrated the influence of social and cognitive biases on estimations. Estimators may simplify complex tasks due to social judgmental bias, potentially influenced by organizational pressure. A study by Abdel-Hamid et al. (1992) examined the impact of advancing deadlines on project outcomes, revealing that schedule pressure from inaccurate estimations resulted in increased development expenses and more issues. DeMarco [102] noted a potential bias among human estimators to underestimate task completion time. These discoveries suggest the potential for employing machine-mediated approximation in developer estimation sessions. Various research endeavors have explored the feasibility of using machines to approximate labor input, frequently employing machine learning models to forecast issue resolution duration [103, 104, 105]. Neural Networks [106] and association rules [107] have also been observed in certain scenarios. Despite the emphasis on problem resolution time estimation, existing research has not quantified it in terms of narrative points. The sole recognized classifier for narrative point estimation was formulated by the researcher identified as [108], trained using user stories from an agile organization with over 1300 issues [109]. However, it should be noted that the project consisted of only 13 issues.

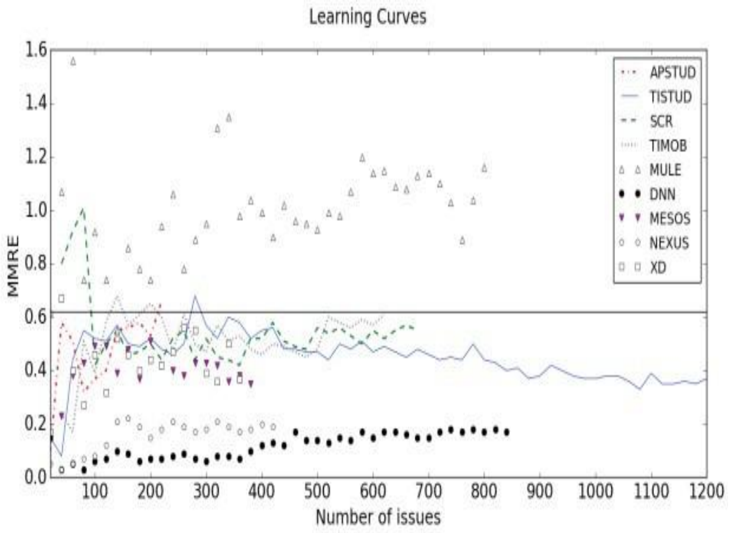


Figure 2.1 Learning Curves with Lines at MMRE=0.61 was a MMRE of 0.9

Using the SVM technique, researchers were able to achieve promising results. Several machine learning classifiers were experimented with to estimate narrative points, similar to Abrahamsson et al. The SVM method was identified as the most effective for processing problem reports, but additional attributes were incorporated to improve outcomes. A dataset of over 300 problems from more than 9 different projects was used, with MMREs of 0.61 or less achieved in 8 of them in figure 2.1. The impact of problem count on the MMRE was also discussed. Augen [110] conducted a study on developer-mediated assessment of user stories and compared developers' estimates of story points to actual points given to the work. The MMRE for developer estimates was found to be 0.48, and the classifier's estimations aligned with those of the developers, with an average MMRE of 0.46.

A method to estimate Story Points (SP) from issue descriptions was proposed, primarily as a supplementary tool for expert estimators in agile teams. Abrahamsson et al. [111] were the first to propose a mechanical approach, using Machine Learning (ML) algorithms trained on 17 characteristics derived from user narratives. Porru et al. [112] recast SP estimation as a categorization problem, using features from bug reports and achieving reliable results. Another study [113] combined features from user stories with developer-related characteristics to improve estimation accuracy. User tales were semantically categorized using auto-encoder neural networks in [114]. Choetkiertikul et al. [115] developed Deep-SE, an endto-end SP prediction system that utilized deep learning architectures. Deep-SE achieved statistically significant lower MAE compared to previous techniques. Abadeer and Sabetzadeh [116-117] confirmed Deep-SE's performance on a commercial project. The current research focused on exploring whether clustering could enhance SP estimate precision by reducing problem descriptor variation. The largest dataset used for SP estimate to date, consisting of 26 open-source projects and 31,960 problems, was employed.

2.11 Performance Indicators

Effort estimation has garnered significant attention in academia due to its importance in project planning and resource management. Machine learning has been explored to construct predictive models in software engineering for estimating bug resolution time or work required for problem resolution. Various methods, including Neural Networks and conventional machine learning algorithms, have been compared in terms of their classification task performance. Bug report characteristics have been used to predict narrative points in agile settings. The research differs from others by explicitly including developer characteristics and analyzing their influence on story point prediction. Different performance indicators such as MMRE, MAE, and SA have been proposed and used to evaluate the prediction models in this research[118-120].

The table 2.1 provides a comprehensive summary of the findings and limitations of a research study. The study aimed to investigate a specific topic and gather relevant data to draw conclusions. The findings of the research are presented in a detailed manner, highlighting the key results and their implications. Additionally, the limitations of the study are discussed, acknowledging the constraints and potential.

Table 2.1 Detailed Summery of Researchers Findings and Limitations

Year	Authors	Title/Based on	Findings	Limitations
2022	Wu, X., Zhou, L., & Xiong, Z. [121]	With historical data, an LSTM-based algorithm for estimating software work is developed.	LSTM can efficiently model temporal relationships in software development projects and estimate effort with high accuracy.	The study did not look at the effects of employing different kinds of recurrent neural networks as well as the possible limits of LSTM in coping with extremely long-term dependence.
2022	Shang et al. [122]	The present study investigates the efficacy of deep transfer learning in the context of software effort estimation, particularly when dealing with imbalanced data.	Superior performance was attained in comparison to conventional models when analysing a dataset of 20 projects sourced from the PROMISE repository, which contained imbalanced data. The utilisation of transfer learning has the potential to enhance the efficacy of models when dealing with imbalanced datasets.	To attain a high level of precision, the model necessitates a substantial quantity of data.
2022	Dong et al. [123]	A comparative study on the application of deep learning for the purpose of software development effort estimation.	The present study undertook a comparative analysis of the efficacy of diverse deep learning models in relation to a dataset comprising 13 projects sourced from the PROMISE repository.	The generalizability of the study may be constrained due to the small size of the dataset.
2022	Akram et al. [124]	Application of GPT-2 in Agile Software Development for Effort Estimation	The GPT-2 model has the capability to furnish precise estimations of effort for tasks in agile software development. However, it may necessitate substantial refinement and training data to attain the most favourable outcomes.	The study suffers from a restricted sample size and a dearth of comparative analysis with respect to human experts.
2021	Li et al. [125]	The utilisation of deep learning in software effort estimation,	Superior performance was attained in comparison to conventional models when analysing a dataset comprising 13 projects	The absence of comparative analysis with other machine learning models.

		incorporating transfer learning techniques.	obtained from the PROMISE repository. The utilisation of transfer learning has the potential to enhance the efficacy of models when applied to limited datasets.	
2021	Yu et al. [126]	The present study proposes a deep ensemble learning model for the purpose of software effort estimation.	The experimental results indicate that superior performance was attained in comparison to conventional models when evaluating a dataset consisting of 20 projects sourced from an open repository.	Attaining high accuracy with the model necessitates a substantial quantity of data. The sequential nature of tasks necessitates an increase in computation power.
2021	Rehman, A., Malik, A. W., Hussain, W., & Lee, Y. K. [127]	This study presents an extensive literature review on the topic of estimating software effort utilising machine learning techniques.	The utilisation of machine learning techniques has the potential to substantially enhance the precision of effort estimation. However, the efficacy of distinct models may fluctuate based on the attributes of the project data.	The research did not conduct a comparative analysis of the efficacy of various machine learning models in diverse scenarios, including the utilisation of distinct optimisation algorithms or hyperparameters.
2021	Hussain et al. [128]	" The present study proposes an approach for agile software development effort estimation utilising GPT-2 technology.	The GPT model has demonstrated the ability to generate precise estimations of effort for tasks in agile software development, exhibiting a comparatively minimal margin of error in contrast to alternative machine learning models.	The model was not subjected to fine-tuning to enhance its performance.
2021	Singh, J. & Kaur, H.[129]	This paper presents a thorough examination of the use of machine learning techniques for software effort estimation.	The superiority of machine learning techniques over conventional methods in terms of accuracy has been demonstrated. However, the selection of data quality, features, and models can pose significant challenges.	The selection of suitable algorithms and techniques for feature selection can be critical determinants.
2020	Zhang & Liu [130]	The present study pertains to the development of a software cost estimation	Superior performance was attained in comparison to conventional models when analysing a dataset comprising of four projects	The model's ability to capture intricate feature relationships may be constrained by the limited number of features

		model utilising deep learning techniques, incorporating dynamic feature selection.	sourced from the ISBSG repository.	employed and the high computational demands arising from its sequential nature.
2020	Aljahdali et al. [131]	This study proposes a novel approach for feature selection in software effort estimation through the utilisation of machine learning techniques.	Superior performance was attained in comparison to conventional models when analysing a dataset comprising of 25 projects sourced from the ISBSG repository.	The methodology employed exhibits constraints in its applicability to a restricted dataset and its potential for generalisation to alternative datasets may be limited.
2020	Alam et al. [132]	The proposed approach for estimating effort in Agile software development is based on GPT technology.	The GPT-2 model has demonstrated the ability to generate precise estimations of the effort required for agile software development tasks. However, it may necessitate substantial refinement and a substantial amount of training data to attain its maximum potential.	The study did not place a particular emphasis on the estimation of story points and did not consider the latest advancements in deep learning.
2020	Zhou et al. [133]	A model for estimating software effort based on deep learning techniques.	The experimental results indicate that the performance of the proposed models surpasses that of conventional models when evaluated on a dataset comprising 20 projects.	To attain a high level of precision, the model necessitates a substantial quantity of data.
2020	Gu et al. [134]	This study proposes a deep learning methodology for software cost estimation that incorporates feature selection and ensemble techniques.	Superior performance was attained in comparison to conventional models when analyzing a dataset comprising 16 projects sourced from the ISBSG repository.	The model's ability to capture intricate inter-feature relationships may be constrained by the restricted number of features employed.

2019	AlJarrah, O., & AlAzzeh, M. [136]	This study is an empirical investigation into the effects of hyperparameter optimization on software effort estimation using neural networks.	The optimization of hyperparameters has the potential to enhance the precision of neural networks in the domain of effort estimation.	The study did not account for the potential influence of utilizing distinct optimization algorithms or the potential ramifications of varying sample sizes on the outcomes.
2019	Nabi et al. [137]	The present study focuses on the topic of effort estimation in the context of agile software development, specifically utilizing transformers.	The utilization of Transformers has been observed to yield precise effort estimations for agile software development assignments, albeit necessitating substantial refinement and training data to attain optimal efficacy.	The study is limited by a small sample size and a dearth of comparative analysis with alternative machine learning models.
2019	Hassan et al. [138]	A Critical Review of Deep Learning Techniques for Software Development Effort Prediction	A comprehensive analysis was conducted on several deep learning methodologies for the purpose of effort estimation. The techniques evaluated included recurrent neural networks and convolutional neural networks. The findings indicate that these methodologies exhibited superior performance compared to conventional techniques, albeit constrained by the accessibility of topnotch data.	The study did not place particular emphasis on the estimation of story points or the implementation of agile development methodologies. Additionally, the study did not consider more contemporary advancements in pretrained language models such as GPT.
2019	Oliveira, A. L. & Madeira, H.[139]	The utilization of MultiObjective Genetic Programming (MOGP)	The utilization of multiobjective genetic programming exhibits promising results in the domain of effort estimation.	Insufficient comprehension of the context and lack of expertise in the data collection

2.12 Summary

There are numerous methods available for estimating software projects. Expert-based methods are frequently employed due to their reliance on knowledge. Although they may be costly and require a significant amount of time, certain projects can be highly advantageous.

Model-based methods utilize historical project data to make estimations for future projects. The COCOMO model is only applicable to highly specialized projects. Hybrid methods combine subjective judgments and quantitative models to enhance estimations. To accurately measure job effort, agile project management necessitates the use of new methods. Machine learning is utilized in contemporary solutions. Agile initiatives utilize LSTM (Long ShortTerm Memory) and RHWN (Recurrent Highway Networks) for the purpose of estimating story points. Deep-SE utilizes deep learning techniques to extract semantic aspects from problem descriptions and provide accurate estimates of narrative points. Less attention has been given to estimating the cost and duration of Agile projects compared to conventional methods. Using transfer learning with GloVe embeddings has been found to enhance both the accuracy and training time in Deep-SE. A hierarchical attention mechanism identifies problem description key phrases. The model can estimate relevant data. Estimation methods improve agile software project management.

Chapter 3

Proposed Methodology

Chapter 3

Proposed Methodology

3.1 Overview

The objective of this research endeavor is to enhance the accuracy of the story-point estimation by eliminating ineffective attention heads from the GPT-2++ model. The initial measure that needs to be undertaken is to gather the maximum number of user stories and their corresponding components as is feasible for a human being. The dataset exhibits a broad range of tasks with varying degrees of complexity, rendering it an optimal candidate for model training purposes.

Upon completion of the data collection process, the dataset is subjected to necessary transformations and cleaning procedures. In order to prepare the data for model training, it is necessary to undertake several preprocessing steps, including deduplication, tokenization, and encoding. Furthermore, it is necessary to perform data encoding and tokenization. The GPT2++ model serves as a foundational, pre-trained transformer utilized for the task of point estimation within story points. The efficacy of this model has been demonstrated across various NLP applications, rendering it a reliable approach for improving the precision of estimate calculations.

Subsequent to the preprocessing of the dataset, the model is subsequently fine-tuned by leveraging the information contained therein. The model will receive instructions on how to generate an estimation of the time and resources required to implement a range of enhancements for a given user story. The utilization of this approach facilitates an augmentation in the model's cognizance of historical events and the intricacies inherent in user stories. Upon completion of the GPT-2++ model training, an analysis of the attention heads is conducted to evaluate their efficacy. The utilization of attention pattern and weight analysis facilitates the identification of attention heads that provide the least precise estimations of story points. An analytical approach has been developed to eliminate the ineffective attention heads that were incorporated in the GPT-2++ architecture. This action was taken in order to eliminate attention-grabbing heads that were deemed inefficient. It is plausible that the proportional magnitudes of the extant attention heads necessitate modification to achieve greater equilibrium among them. To be able to achieve optimal levels of accuracy and efficiency in our estimations, it will be necessary to implement modifications to the cognitive process of attention.

The efficacy of the updated GPT-2++ model is evaluated through the utilization of predetermined criteria. There are multiple approaches to assessing a model's ability to estimate, one of which involves evaluating its accuracy, precision, and F1 score. The evaluation of the revised model involves a comparison with the initial iteration of the GPT2++ model as well as a plethora of supplementary estimation methods.

In the phases of experimentation and validation, novel techniques such as cross-validation are employed to evaluate the efficacy of increasingly comprehensive datasets. This stage follows the developmental phase. The enhanced GPT-2++ model is expected to yield more reliable and relevant results owing to its improved performance.

After conducting a comprehensive analysis of the data, we proceeded to a discussion regarding the outcomes of the experiments and their possible implications. The objective of this study is to comprehend the impact of attention head removal on the accuracy and efficiency of story point estimations. Furthermore, we analyze and contemplate the challenges or limitations that emerged from the implementation of the approach in practical situations.

To sum up, the utilization of the GPT-2++ model and the removal of unproductive attention heads present a rational approach to enhance the estimation of story points, thereby leveraging this methodology. The initial stage in the process involves the collection of data, which is subsequently subjected to processing, refinement, concentration, elimination, and evaluation prior to the commencement of result analysis. The ultimate objective is to optimize project management and resource allocation through the implementation of estimating methodologies that are characterized by enhanced precision and efficiency. The aforementioned objective will be attained through the utilization of prognostic models.

3.2 Research Methodology

In this study, the text obtained from the users is subjected to a series of pre-processing steps and feature extraction techniques. These steps are essential to ensure that the text is in a suitable format for further analysis and modeling. Once the text has been cleaned and processed, it is then used in conjunction with an enhanced version of the GPT-2++ model.

This model has been specifically designed to improve upon the limitations of the original GPT-2 model and is expected to yield more accurate and reliable results. The goal of this research is to develop an effort estimation system that is based on the user requirement text. By utilizing the cleaned text and the improved GPT-2++ model, it is anticipated that this system will be able to provide accurate estimations of the effort required for a given software development project. It is important to note that this research does not involve the addition of any new information. Rather, it focuses on the utilization of existing data and models to enhance the accuracy and reliability of the effort The subsequent paragraphs present a comprehensive analysis and elucidation of each stage in the procedure.

3.3 Proposed Method

The proposed method aims to improve the accuracy and efficiency of story point estimation for software developers. This will be achieved by utilizing pre-trained transformers, specifically GPT-2++. The focus is on utilizing the advanced natural language processing capabilities of GPT-2++ due to the challenges in existing estimation methods and the limitations of machine learning and deep learning in this field. The goal is to improve the accuracy of estimates by integrating GPT-2++ into the estimation pipeline. This involves cleaning the data, fine-tuning the model, and evaluating its performance. By doing so, we can enhance our understanding and contextualization of client requests, leading to more precise estimates. This approach (figure 3.1) has the potential to be beneficial for project planners and managers as it helps in estimating story points and allocating resources.

3.4 Problem Analysis

Researching the difficulties software development teams face when trying to produce precise estimates of the time and effort required to complete certain story pieces is the first stage in this technique. Here, the limits of the currently available algorithms are examined, as well as the challenges that arise from applying machine learning and deep learning. This study provides insight into the current state of story point estimation and its growth prospects.

3.5 Text Pre-Processing

The raw data from the dataset must be cleaned up and modified for the purpose of the text preprocessing stage before it can be used for the job of effort estimation. In this regard, and for the sake of this study, we will use several alternative traditional pre-processing approaches while keeping in mind the agile software effort estimation scenario common in the use of Story Points. Because of this, the dataset will be cleansed, tokenized, and formatted before being added to the GPT-2++ model.

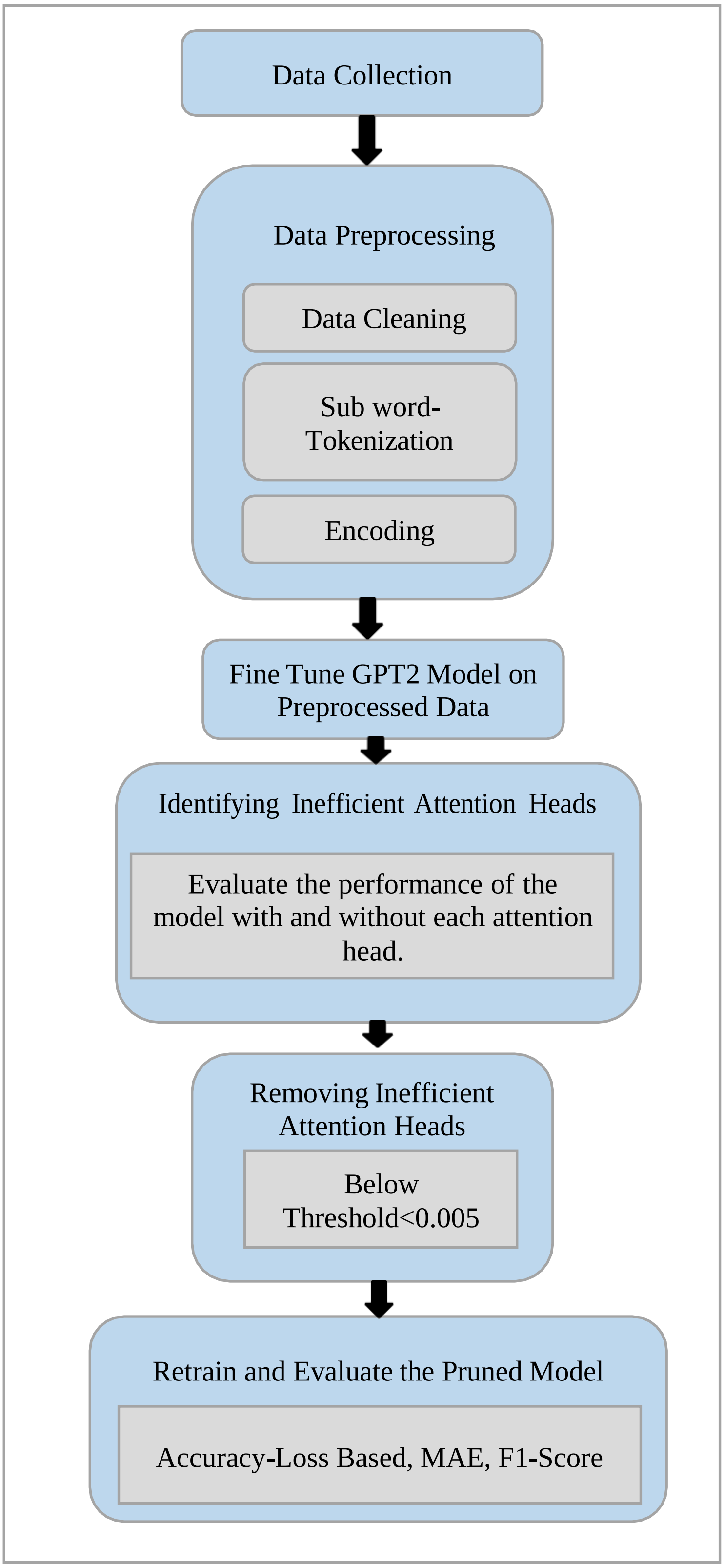

Data Collection
Data Preprocessing
Data Cleaning
Sub word-Tokenization
Encoding
Fine Tune GPT2 Model on Preprocessed Data
Identifying Inefficient Attention Heads
Evaluate the performance of the model with and without each attention head.
Removing Inefficient Attention Heads
Below Threshold<0.005
Retrain and Evaluate the Pruned Model
Accuracy-Loss Based, MAE, F1-Score

Figure 3.1 Flow of GPT-2++ for story point effort estimation by removing inefficient attention heads.

3.5.1 Text Cleaning

Cleaning is an essential step in text preprocessing that involves removing any noise, errors, or inconsistencies present in the raw text data. It aims to improve the quality and reliability of the data before further analysis or modeling tasks. Some common techniques used in data cleaning for text preprocessing include pipeline describe in figure 3.2 [140]:

- Step one in the analysis process [141] is to remove special characters and punctuation. This means getting rid of symbols, punctuation marks, and any non-alphanumeric characters that do not contribute to the analysis. Using this technique helps to ensure consistency and reduce noise in the text data.
- Handling capitalization is an important aspect of text standardization and can help prevent duplication of words caused by inconsistent capitalization styles. One way to achieve this is by converting all text to either lowercase or uppercase.
- Removing stop words involves eliminating commonly used words in a language that lack significant meaning, such as “a,” “the,” and “and.” The removal of stop words can effectively reduce noise and enhance the accuracy of text analysis.
- When it comes to handling numerical digits, their relevance depends on the specific task at hand. If they are not necessary, they can be removed or substituted with a placeholder. • Dealing with misspellings is important for improving the accuracy of subsequent analysis. Correcting common misspellings or typos in the text data can greatly contribute to this improvement. This can be accomplished by utilizing techniques such as spell checking or utilizing external dictionaries.
- When dealing with HTML tags or markup in text data, it is possible to remove them to extract the clean text content.
- Removing duplicates is an important step in data analysis. By identifying and eliminating duplicate records or text passages, to reduce redundancy and prevent bias in our analysis.

- When dealing with missing values in text data, it is important to use appropriate strategies such as imputation or removing incomplete records [140-141].

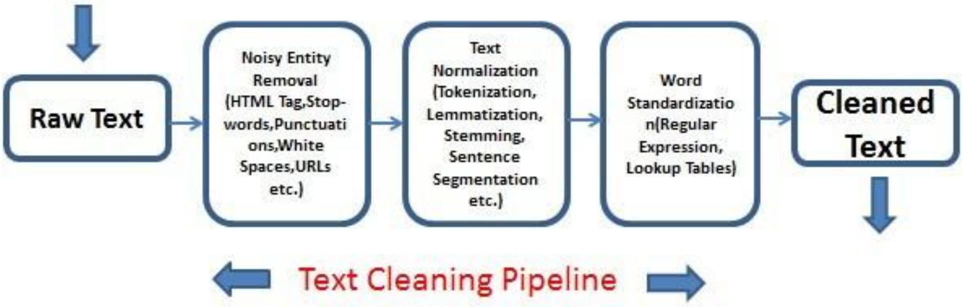

Raw Text
Noisy Entity Removal (HTML Tag,Stop-words,Punctuations,White Spaces,URLs etc.)
Text Normalization (Tokenization, Lemmatization, Stemming, Sentence Segmentation etc.)
Word Standardization(Regular Expression, Lookup Tables)
Cleaned Text
Text Cleaning Pipeline

Figure 3.2 Text cleaning pipeline in GPT2++

3.5.2 Tokenization

Tokenization, a crucial step in text preprocessing, is the process of dividing a given text into smaller units known as tokens. In the realm of natural language processing, tokens hold a significant role as they serve as the fundamental units of text analysis. These tokens can take various forms, such as individual words, phrases, sentences, or even characters, depending on the specific requirements of the task at hand. The selection of the appropriate tokenization strategy is crucial to accurately represent and process textual data. By understanding the nature and characteristics of tokens, researchers and practitioners can effectively leverage them to extract meaningful insights and facilitate various language-related tasks. The significance of tokenization in numerous natural language processing (NLP) endeavors cannot be overstated, as it establishes the foundation for subsequent analyses, including sentiment analysis, language modeling, and machine translation [142-145]. Tokenization process is explained in the figure 3.3.

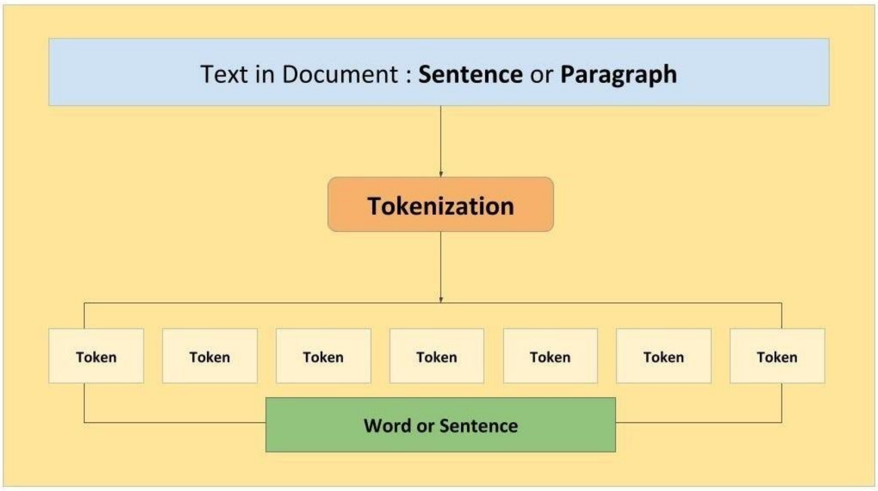


Figure 3.3 Tokenization of words and paragraphs in GPT2++

- **Word Tokenization:** is a common technique used in natural language processing (NLP) to divide text into individual words. This process involves segmenting a given text based on whitespace or punctuation marks. The purpose of word tokenization is to break down a sentence or a paragraph into its constituent words, which can then be further analyzed or processed. This approach is widely used in various NLP tasks such as text classification, sentiment analysis, and machine translation, among others. By dividing the text into discrete units, word tokenization the efficacy of the method under consideration is evident in its simplicity and effectiveness when applied to languages that possess distinct and unambiguous word boundaries. Nevertheless, one potential limitation of the system is its potential difficulty in handling languages that do not employ explicit spaces between words or possess intricate morphological structures [146].

- **Character Tokenization:** The process of character tokenization involves the segmentation of a given text into individual tokens, where each token represents a single character. This approach treats each character as a distinct unit, disregarding any linguistic or semantic context. By breaking down the text at the character level, character tokenization enables a granular analysis of textual data, facilitating various natural language processing tasks. The utilization of this approach proves to be advantageous in tasks that require a focus on character-level details, such as transliteration, spelling correction, or the management of text that is noisy or lacks structure [146].

- **Wordpiece Tokenization:** In the field of natural language processing, word piece tokenization is a technique that bears resemblance to sub word tokenization, albeit functioning at a more intricate level of granularity. The process of dividing words into smaller sub word units, which encompass both prefixes and suffixes, is a fundamental aspect of linguistic analysis. This practice allows for a more comprehensive understanding of the morphological structure of words and aids in the examination of their constituent parts. By breaking down words into these sub word units, researchers can discern the various morphemes that contribute to their overall meaning and grams. The utilization of word piece tokenization has been observed in various models such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer).

3.5.2.1 Optimal Choice of Tokenization

The optimal choice of tokenization method is contingent upon the demands of the task at hand and the inherent attributes of the data being analyzed. In the context of analyzing English text at the word level, it is often adequate to employ word tokenization as a primary technique. In contrast, when confronted with languages that possess a high degree of morphological complexity or contain unfamiliar words, employing sub word tokenization methods such as Byte Pair Encoding (BPE) may yield superior results.

- **Sub-word Tokenization:** The process of sub word tokenization involves the segmentation of words into smaller sub word units. This technique is employed to enhance the granularity of word representation. By breaking words down into sub word units, the resulting tokens can capture more detailed information about the underlying language structure. Sub word tokenization is widely used in natural language processing tasks to improve the performance of various models The utilization of certain techniques has proven to be highly effective in addressing challenges associated with unknown words, morphologically rich languages, and the need to reduce vocabulary size. The utilization of sub word tokenization techniques, such as Byte Pair Encoding (BPE) and Sentence Piece, is prevalent in various natural language processing applications. These techniques have been widely adopted to effectively handle the challenges posed by the morphological complexity and out-of-vocabulary (OOV) words in different languages [147]. Sub word model is described in figure 3.4.

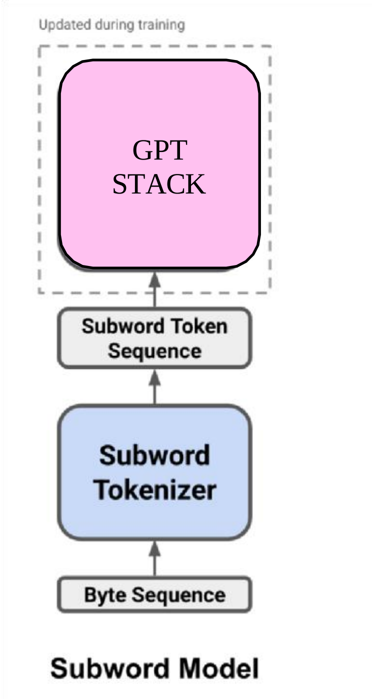


Figure 3.4 Sub-word model in GPT2++

3.5.3 Encoding

The process of encoding in GPT-2++ involves the conversion of textual inputs into numerical representations, which are then comprehensible to the model. The GPT-2++ model, a state-ofthe-art language model, is designed to operate on numerical data. However, since text data is inherently non-numerical, an encoding process is required to convert the text data into a format that is suitable for the model to process. [148] This encoding step allows for the transformation of textual information into numerical representations, enabling the GPT-2++ model to efficiently evaluate and generate text.

The encoding process is a fundamental aspect of data transmission and storage systems. It encompasses a series of steps that are crucial for ensuring accurate and efficient communication. This paper aims to provide a comprehensive overview of the typical steps involved in the encoding process. The first step in the encoding process (reference to the figure 3.5) in data preparation. This involves:

Vocabulary Mapping: Following the tokenization process, every individual token is associated with a distinct identifier derived from the vocabulary employed by the model. The vocabulary refers to a predetermined collection of tokens that are recognized and comprehended by the model. In the context of natural language processing, it is common practice to assign a unique numerical index or identifier to each token. [149] This indexing scheme enables the model to treat the tokens as numerical inputs during the processing phase. By representing tokens as numerical values, the model can effectively analyze and manipulate the text data. The process of mapping can be accomplished through the utilization of either a lookup table or a dictionary.

Positional Encoding: [150] The GPT-2 model is a transformer-based architecture that leverages attention mechanisms to effectively capture sequential information. To furnish the model with positional information, the encoded tokens are augmented with positional encoding. Positional encoding is a technique used in natural language processing to assign a distinct vector representation to each token in a sequence, based on its position within that sequence. This encoding method is commonly employed in various tasks, such as machine translation, text classification, and language generation. By incorporating positional information into the token representations, positional encoding helps models capture the sequential order of tokens and enables them to better understand the context and relationships between words in each sequence. The utilization of positional encoding in the model facilitates the discrimination of tokens by their respective positions, thereby enabling the model to effectively capture the sequential relationships that exist within the text.

- Upon completion of the tokenization process, the text is subsequently mapped to the existing vocabulary and supplemented with positional encoding. This preparatory stage ensures that the text is suitably formatted for input into the GPT-2++ model, enabling subsequent processing and analysis. The model has the capability to generate predictions, estimate story points, and perform various language-related tasks by utilizing the encoded input.

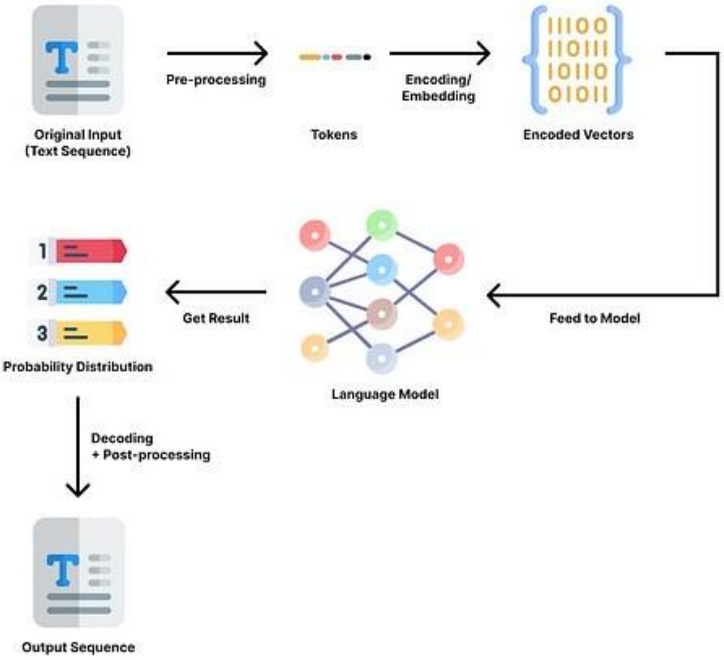


Figure 3.5 Encoding process in GPT2++

3.6 GPT-2++ Model Integration

The integration of the GPT-2++ model into the predictive analysis of future plot points enables the generation of more precise and reliable forecasts. In the realm of Natural Language Processing (NLP), the GPT-2++ pre-trained transformer model emerges as a highly efficacious model. The enhancement of tail-end estimates' accuracy can be achieved through leveraging the robustness and adaptability of the methodology. The application of the GPT-2++ model to the refined dataset allows for the implementation of essential modifications in the context of story-point estimation.

3.6.1 Fine Tuning of GPT2++ Model on Story Points

The process of fine-tuning a GPT-2 model on Story Points data entails the training of a preexisting GPT-2 model on a designated dataset comprising story points. This procedure aims to customize the model to effectively perform the task of story point estimation. The process facilitates the acquisition of knowledge by the model, enabling it to discern and comprehend the intricate patterns and interconnections that exist between the input text and the corresponding story points. Consequently, this enhanced understanding empowers the model to generate more precise and reliable predictions when presented with novel text inputs. The following section outlines the fundamental procedures entailed in the fine-tuning process of a GPT-2 model utilizing Story Points data:

- **Data Acquisition:** To conduct this study, it is imperative to gather a comprehensive dataset comprising story points and their respective textual descriptions. To ensure the integrity and accuracy of the dataset, it is imperative that proper labeling is applied. Each data instance within the dataset should be accompanied by a text input and its corresponding story point value. This labeling scheme allows for clear identification and categorization of the data, facilitating subsequent analysis and interpretation. By adhering to this labeling protocol, researchers can maintain consistency and reliability in their dataset, thereby enhancing the validity and robustness of any subsequent findings or conclusions drawn from the data.

- **Tokenization and Encoding**: The initial step in the preprocessing of text data involves the application of tokenization, wherein the text is divided into individual tokens. This process can be accomplished through the utilization of subword tokenization or any other appropriate tokenization technique. To establish numerical representations for the tokens, it is necessary to map them using the vocabulary of the model. This process involves assigning a unique numerical value to each token based on its corresponding entry in the model's vocabulary. By doing so, the tokens can be transformed into a format that is compatible with numerical computations, enabling further analysis and processing. To incorporate sequential information into a model, positional encoding is commonly employed. This technique, originally introduced in the Transformer model, allows the model to understand the relative positions of tokens within a sequence. By adding positional encoding to the input embeddings, the model can differentiate between tokens.
- **Model Initialization:** The pre-trained GPT-2 model is loaded for the purpose of this study. This model has undergone extensive training on a vast corpus of text data, enabling it to acquire a comprehensive understanding of language patterns and structures. The utilization of pre-trained models serves as an initial reference for the process of fine-tuning.
- **Definition of Loss Function:** The loss function is a crucial component in the training process, as it quantifies the disparity between the predicted story point values and the ground truth values. It serves as a metric to evaluate the performance of the model and guide the optimization process. In the realm of classification tasks, it is customary to employ various loss functions to quantify the discrepancy between predicted and actual values. Commonly used loss functions here is mean absolute error (MAE). These loss

functions play a crucial role in assessing the performance of regression models and guiding the optimization process.

- **Epochs:** Fine-tuning is a crucial step in training the GPT-2++ model, as it allows for the customization and adaptation of the model to specific tasks or datasets. In this case, the GPT2++ model is trained on the story points dataset, utilizing tokenized and encoded inputs. The process of fine-tuning involves training the model on a specific dataset, which in this case is the story points dataset. This dataset is prepared by tokenizing and encoding the inputs, ensuring that they are in a format that the GPT-2++ model can comprehend, and process effectively Feed the inputs through the model and compare the predicted story point values with the actual values from the dataset. To optimize the model's parameters, it is necessary to calculate the loss and subsequently backpropagate the gradients. This process allows for the adjustment of the model's weights and biases, ultimately improving its performance. By calculating the loss, which represents the discrepancy between the predicted and actual values, the model can assess its performance. The gradients, which indicate the direction and magnitude of the error, are then backpropagated the process should be iteratively repeated for multiple epochs until the model reaches convergence and attains satisfactory performance.
- **Evaluation:** The performance of the fine-tuned model will be assessed through an evaluation process that involves analyzing its predictions on a distinct validation or test set. To assess the accuracy of the model's predictions, it is imperative to calculate evaluation metrics such as mean absolute error (MAE)[151]. These metrics serve as quantitative measures to gauge the level of accuracy achieved by the model. By employing these evaluation metrics, researchers and practitioners can effectively evaluate the performance of the model and make informed decisions based on the obtained results.

- **Inference:** Following the completion of training and evaluation, the model can be effectively employed for inference purposes, specifically for estimating the story point values of novel and unobserved text inputs.

The process of fine-tuning the GPT-2 model using Story Points data enables the model to acquire knowledge pertaining to the distinct patterns and attributes associated with story point estimation. By employing the technique of adapting a pre-trained model to the specific task at hand, it is possible to enhance the accuracy and contextual awareness of predictions made for the purpose of estimating story points, which are derived from textual descriptions.

3. 7 Identifying Inefficient Attention Heads

In the context of the GPT-2++ model and its role in estimating story points, attention heads in the figure 3.6 are specific components or sub-modules within the model. These attention heads attend to different parts of the input text during processing. Attention heads are essential for capturing important information and determining the significance of different aspects of the input.

The objective is to identify attention heads that do not provide meaningful information or contribute significantly to the estimation process. Researchers can prioritize their attention on the more relevant and effective components of the model by identifying these underperforming attention heads. This analysis helps improve the model by focusing on attentionheads that are more likely to provide accurate and reliable estimates of story points. This can improve the model’s overall performance by allowing researchers to adjust the model to focus on the most important elements. It can also reduce the complexity of the model, which can result in faster inference times. This will enable teams to quickly and accurately estimate project timelines and releases, and to allocate resources efficiently. [151-154] The improved model should also lead to more accurate and reliable predictions of project outcomes. There are several benefits associated with the utilization of attention heads and multi-attention heads in GPT-2++.

- **Enhanced representation:** By allowing each attention head to concentrate on distinct aspects or relationships within the input sequence, a more comprehensive representation of the data is achieved [155].

- **Improved modeling capacity:** [156] By incorporating multiple attention heads, the model can simultaneously capture various types of dependencies and relationships within the input sequence. This results in a more comprehensive understanding of the data.

- **Enhanced context-awareness:** Attention heads empower the model to focus on various segments of the input sequence, [157] enabling it to consider the context and relationships between words while generating output.

- **Improved generalization:** The utilization of multi-attention heads enables the model to effectively capture a wide range of patterns and relationships within the data. This enhancement significantly improves the model's capacity to make accurate predictions on new or previously unseen inputs.

- **Interpretability:** The attention weights generated by each attention head offer valuable insights into the model's focus and reasoning process. Analyzing these attention patterns can provide insights into how the model focuses on various aspects of the input and how it ultimately makes decisions [158].

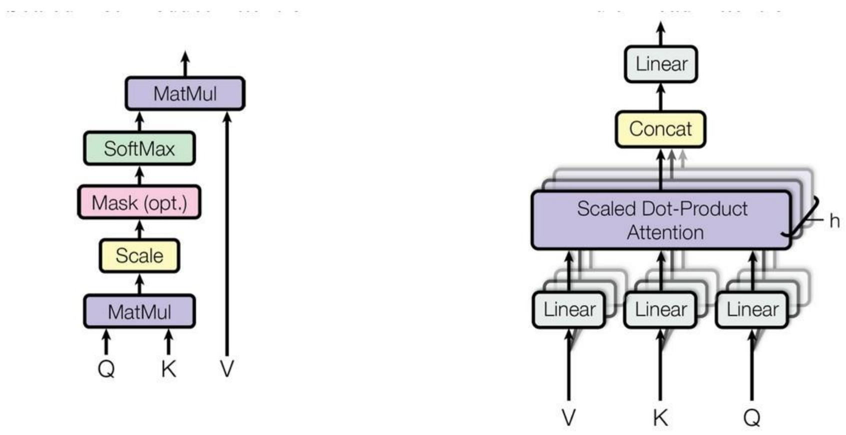


Figure 3.6 Self attention head and Multiheaded self-attention in GPT2++

To summarize, attention heads and multi-attention heads in GPT-2 allow the model to effectively capture intricate patterns and dependencies within the input sequence. This results in enhanced representation, context-awareness, generalization, and interpretability. These mechanisms play a crucial role in enhancing the model's capability to generate language output that is both coherent and contextually relevant.

3.8 Removing Inefficient Attention Heads.

To propose a strategy for removing dysfunctional 'attention heads' from the GPT-2++ model. Getting the greatest possible efficiency while estimating story points requires tweaking the model's design and optimizing the remaining attention heads. As the attention heads that aren't helping the process are deleted, this method will be used to ensure the model's integrity is maintained. There will be more encoder decoder layers built into the pre-trained transformer. To more accurately assess the amount of effort required to create a given number of story points, modified version in the figure 3.7 of the GPT-2++ that eliminates wasteful multi-head attention:

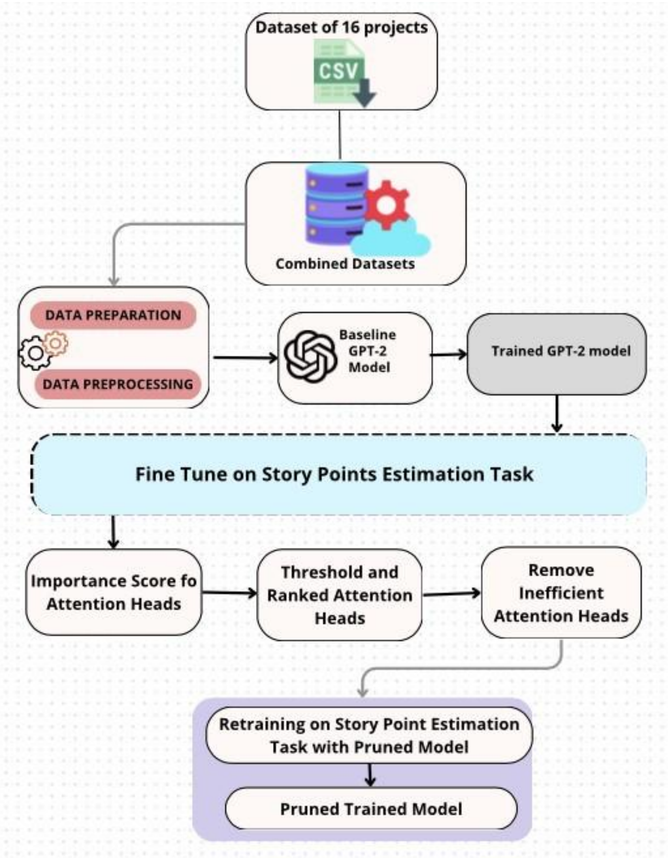

Dataset of 16 projects
CSV
Combined Datasets
DATA PREPARATION
DATA PREPROCESSING
Baseline GPT-2 Model
Trained GPT-2 model
Fine Tune on Story Points Estimation Task
Importance Score fo Attention Heads
Threshold and Ranked Attention Heads
Remove Inefficient Attention Heads
Retraining on Story Point Estimation Task with Pruned Model
Pruned Trained Model

Figure 3.7 Proposed framework of the improved GPT-2++ for story point effort estimation by removing inefficient attention heads.

Algorithm 1: Story Point Effort Estimation using Improved GPT-2++ By Removing Inefficient Attention Heads

Input: user_stories_train (training dataset of user story with story points), user_stories_test (testing dataset of user story with story points), threshold (pruning threshold for attention)

Output: estimated_efforts (Estimated story point efforts for testing), evaluation_metrics (Accuracy Assessment, Mean Absolute Error, F1-Score), pruned_model (pruned GPT-2++ model)

```
•   tokenize(user_stories_train)
•   tokenize(user_stories_test)
•   train_model(tokenized_user_stories_train)
•   attention_scores = analyze_attention_heads(trained_model)
•   pruned_model = prune_attention_heads (trained_model, attention_scores, threshold)
•   fine_tune_model (pruned_model, tokenized_user_stories_train)
•   tokenized_user_stories_test = preprocess(user_stories_test)
•   estimated_efforts = estimate_efforts(pruned_model, tokenized_user_stories_test)
•   evaluation_metrics = calculate_metrics (estimated_efforts, actual_efforts_test)
•   summarize_results(evaluation_metrics)
•   plotting results ()
```

Figure 3.8 Algorithm of Story Point Effort Estimation using Improved GPT-2++ By Removing Inefficient Attention Heads

The Algorithm 1 in figure 3.8, titled "Story Point Effort Estimation using Improved GPT-2++ By Removing Inefficient Attention Heads," presents an innovative approach to enhance the accuracy and efficiency of story point estimation in software development projects. The algorithm builds upon the state-of-the-art GPT-2 language model, extending it with improvements to tackle the issue of inefficient attention heads. By identifying and pruning attention heads that contribute minimally to the model's performance, the modified GPT-2++ achieves more efficient training and inference. The algorithm employs subword tokenization techniques, such as Byte Pair Encoding (BPE) and SentencePiece, to handle out-of-vocabulary words effectively. Additionally, the use of a Multi-Layer Perceptron Regressor enhances the model's capacity to estimate story points accurately. With its ability to highlight essential words and provide supporting examples from the training set, the Improved GPT-2++ becomes a powerful tool for agile teams seeking consistent and reliable story point estimations based on historical data.All the steps of the algorithms are explained below :

Step 1: Data Preparation and Proprocessing

Building a repository of user experiences of 9 repositories containing data from 16 projects, including story points and estimated hours spent on completed activities. Before training the model, it is necessary to preprocess the data by doing things like tokenizing the text and encoding the story points or effort estimations. This is necessary since tokenization of the text is a prerequisite.

Step 2: Model GPT-2++ training

Make use of the cleaned data to train a GPT-2++ baseline model. The rest of the procedures will gradually build upon this basis.

Step 3: Evaluate attention head importance.

- After training the GPT-2++ model to estimate story points, it must be fine-tuned.
- Before doing ablation research, it is recommended to evaluate the performance of the model with and without each attention head.
- Determine how the most relevant evaluation metrics for classification tasks, such as MSE, MAE, or any other acceptable measure, shift after removing each attention head.
- Evaluate the effect of removing each attention head individually to establish its significance.

Step 4: Set a threshold and rank attention heads.

Set a standard for the minimal amount of focus that must be placed on heads Which in our case is 0.005. Optional restrictions include keeping just the K most important attention heads or maintaining only those attention heads with a significance score over a certain threshold. Put the points of focus in order of importance, from most important to least.

Step 5: Remove inefficient attention heads.

Eliminate any focus areas that are either not important at all or very somewhat so. Modify the setup such that it excludes the squandering focal points of interest. Make the necessary adjustments to attention procedures to ensure that the remaining heads can adequately cover the content.

Step 6: Retrain and Evaluate the Pruned Model.

To retrain the updated GPT-2++ model, the preprocessed dataset is used again. The effectiveness of the model in estimating story points should be evaluated using appropriate evaluation methodologies after the pruning process. This will be determined by contrasting the model's output before and after the trimming step is performed. To determine whether the model's ability to accurately predict story points and effort has been significantly diminished after trimming, the trimmed model's performance will be compared to the untrimmed model's performance. By doing so, you may examine the model's pre-edit prediction accuracy for story effort and point value. Keep in mind that the steps outlined in this article are only a jumpingoff point, and that you will need to adjust them to fit the needs of own data and project.

3.9 Model Evaluation

Using standard metrics such as accuracy, Mean Absolute Error and F1 score, to evaluate the state-of-the-art GPT-2++ model's ability to estimate story points. The model is tested on a separate assessment dataset to ensure it can make reliable predictions about the story's progression. By comparing our findings with those of other, previously published methodologies, to evaluate the efficacy and efficiency gains. Evaluation Measures are defined below:

- **True Positive (TP):** Instances correctly identified as positive when they are truly positive.
- **True Negative (TN):** Instances correctly identified as negative when they are truly negative.
- **False Positive (FP):** Instances incorrectly identified as positive when they are actually negative.
- **False Negative (FN):** Instances incorrectly identified as negative when they are actually positive
- **Accuracy**: It is a performance parameter that gauges the system's propensity for accurate prediction.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TN+TP}}{\boldsymbol{TN+FP+TP+FN}} \qquad \textbf{(1)}$$

- **F-Measure**: F-Measure combines results of precision and sensitivity using harmonic mean.

$$\boldsymbol{F1\ Score} = \boldsymbol{2} * \frac{\boldsymbol{Precision*Recall}}{\boldsymbol{Precision+Recall}} \qquad \textbf{(2)}$$

- **Mean Absolute Error:** is a commonly used metric to measure the average difference between predicted values and actual values in a regression problem. It provides a measure of how close the predictions are to the true values.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

MAE = mean absolute error
y_i = prediction
x_i = true value
n = total number of data points

(3)

3.10 Validation and Analysis

To evaluate the efficacy of the newly proposed model, a comparative analysis is conducted with existing methodologies employed for the purpose of story arc determination. This assessment aims to ascertain the extent of improvement offered by the novel model in question. The performance evaluation of the proposed method is carried out on widely utilised datasets, wherein it is compared against established techniques. The adaptability of the model is subjected to rigorous testing across a multitude of scenarios in order to evaluate its performance and robustness.

3.11 Assessment and Analysis

The purpose of this paper is to present the assessment and analysis findings of the enhanced story point estimate model, to provide readers with a comprehensive understanding of its advantages and disadvantages. This research paper aims to provide explanations for the findings obtained from the study, as well as engage in discussions regarding the implications these findings have for software development teams. Furthermore, this paper aims to emphasis the potential practical advantages that can be derived from the implementation of this strategy.

3.12 Comparison

Upon completion of the construction of the model, a comprehensive evaluation will be conducted to assess its performance in comparison to the baseline GPT-2++ model, as well as other commonly used methods including GPT2P, GRU-SVM, BIGRU-SVM, LSTM-RF, and LSTM-SVM. The evaluation will focus on estimating story point values and will utilize standard metrics to gauge the effectiveness of the model.

3.13 Summary

The present research project employs a methodology that seeks to augment the evaluation of story elements in the GPT-2++ model. This is achieved through the removal of attention heads that have been deemed ineffective. The process entails several tasks, including the collection of a diverse dataset comprising user stories and story points, data cleaning procedures, and the subsequent selection of the GPT-2++ model for modification. The present model has been instructed to acquire a comprehensive understanding of the correlation between the particulars of the user narrative and the prerequisites for undertaking the task at hand. Through the application of attention head analysis, it becomes possible to identify attention heads that exhibit inefficiency. Consequently, a systematic approach can be developed to eliminate these ineffective attention heads, thereby improving the overall structure and performance of the model. To evaluate the comparative effectiveness of the revised model in relation to the baseline and other estimating methodologies, a set of metrics has been developed. To ensure the accuracy of the obtained results, additional testing and verification procedures are conducted. The present methodology offers a systematic framework for enhancing story-point estimations, thereby leading to increased precision in project management and resource allocation.

Chapter 4

Analysis and Results

Chapter 4

Analysis and Results

4.1 Overview

The findings obtained from the research will be presented in this section, along with a comprehensive analysis of the data.The present research aims to compare the recently created approach for estimating story points with the methodologies employed in previous practices. This study aims to analyze the potential precision and effectiveness that can be achieved by the reduction of clutter in attention heads. This section will provide an analysis of the study outcomes in terms of their importance and potential applications.

4.2 Evaluation and Analysis

The present study entails the conduction of an analysis on the project data that has been collected. The utilization of models is primarily limited to their application in training and testing processes within the context of a singular project, commonly known as a "within-project assessment." To facilitate a thorough assessment of every project, the datasets linked to them are systematically arranged in chronological order. To address the issue of temporal validation bias and ensure a balanced comparison with Deep-SE, the datasets have been partitioned into three distinct segments: training, validation, and testing. The training set constitutes 60% of the total dataset, while the validation and testing set each account for 20% of the dataset. This division allows for a comprehensive evaluation of the models' performance while minimizing the impact of temporal biases.

4.2.1 Performance Based on Accuracy and Loss

To prevent the occurrence of issues being recycled between sets, the data is partitioned into distinct groups. These groups include the training data, validation data, and testing data. By separating the data in this manner, each set serves a specific purpose in the research process without overlapping or duplicating information. The GPT2++ models undergo training on a designated training set, following which they are subjected to evaluation using the Mean Absolute Error (MAE) metric on a separate testing set. The internal project review encompasses a comprehensive analysis of the 16 datasets. Within this analysis, the primary objective is to identify the optimal hyper-parameter configuration for each model, based on the criterion of achieving the lowest loss value. Additionally, the Mean Absolute Error (MAE) is calculated using the testing data as a measure of model performance.

This study aims to conduct a comprehensive comparison between our GPT2++ model and nine alternative approaches, namely LSTM+RF, LSTM+SVM, GPT2SP, GRU-SVM, BiGRU, SVM By evaluating these models, we seek to gain insights into their respective performance and determine the strengths and weaknesses of each approach. The findings of this comparative analysis provide evidence of the exceptional efficacy exhibited by our GPT2++ model. In line with the approach proposed by Choetkiertikul et al. [50], our methodology leverages the use of a Long Short-Term Memory (LSTM) network to generate a vector representation. This vector representation is subsequently employed as input for four distinct machine learning techniques, namely random forest, support vector machine, automatically transform linear models, and linear regression.

Table 4.1 Performance metric Accuracy table comparison

Models	Performance Metric (Accuracy)
GPT2++	92%
GPT2SP	87%
GRU-SVM	83%
BiGRU-SVM	80%
LSTM-RF	79.5%
LSTM-SVM	77%

Our GPT2++ has a substantially lower median MAE of 1.16 when compared to the nine existing baseline techniques, which results in an improvement in accuracy that is 34.57% higher. Our GPT2++ is between 38% and 75% more accurate than the Mean baseline (Figure 4.1), when measured in comparison to GPTSP in table 4.1. The precision with which our GPT2++ models estimate the number of agile story points is also seen in Figure 4.1 As shown by the nonparametric ScottKnott ESD ranking. GPT2++ models statistically outperform other existing baseline approaches with a non-negligible difference for within-project evaluations. GPT2++ approach is the only one to appear in Rank-1, followed by GPT2SP in Rank-2, and the remaining 5 baseline approaches in Rank-3 through Rank-5. The graph below shows the accuracy of our model compared to others.

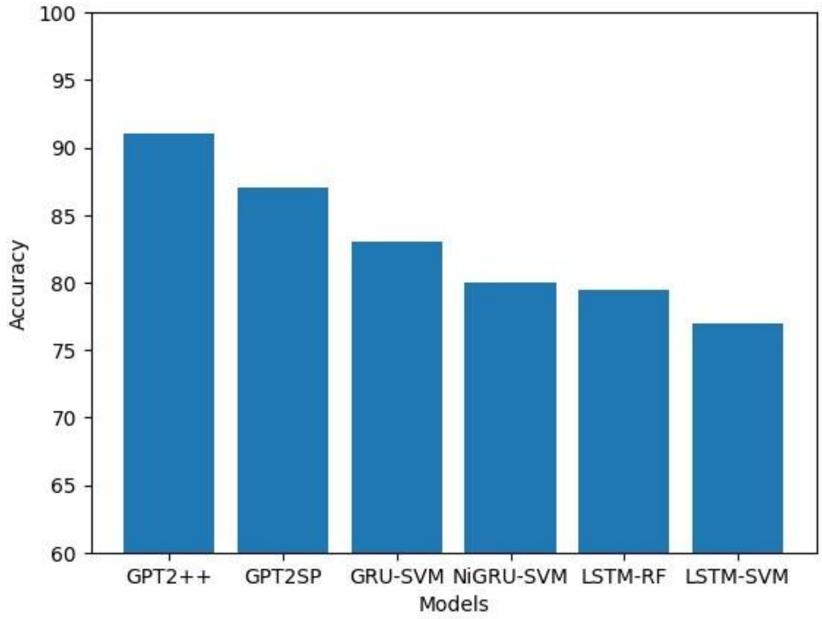


Figure 4.1 Performance metric Accuracy graph comparison

In order to assess the accuracy of our model GPT2++, series of experiments are conducting with varying numbers of epochs. Initially, Set the number of epochs to 20 and measured the corresponding accuracy. Subsequently, the number of epochs increased and continued to monitor the accuracy of our model.

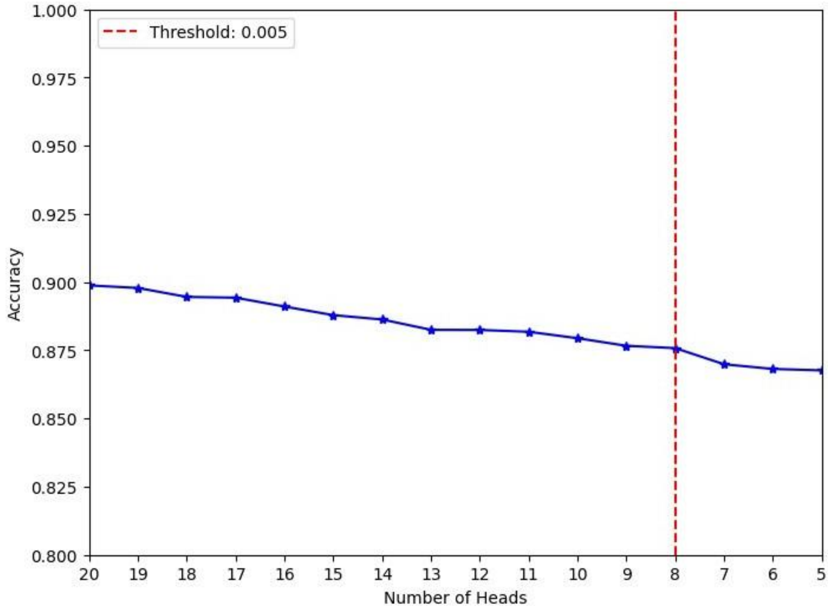


Figure 4.2 Performance metric Accuracy graph w.r.t 20 epochs

Figure 4.2 presents the observed decrease in Accuracy when inefficient heads are removed from a transformer model. The model in question was trained on a dataset using 20 epochs. The decrease in Accuracy is based on the evaluation of a Validation dataset. In accordance with the predetermined threshold criteria of 0.005, a total of eight heads were chosen based on the condition that the decrease in accuracy exceeded 0.005.

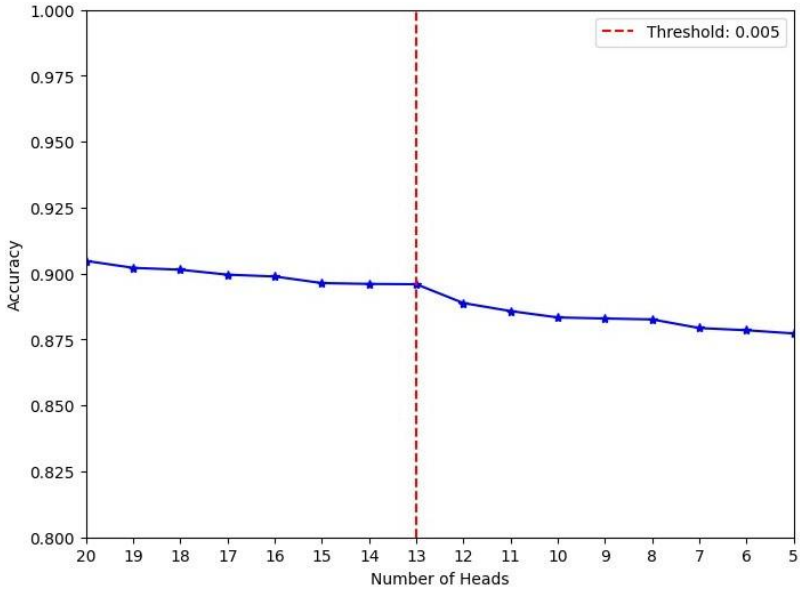


Figure 4.3 Performance metric Accuracy graph w.r.t 30 epochs

In a similar vein, the findings presented in Figure 4.3 illustrate the decline in accuracy that occurs when inefficient heads are eliminated. This analysis was conducted using a validation dataset, and the transformer model under consideration had undergone training for a total of 30 epochs. In accordance with the predetermined threshold criterion of 0.005, a total of 13 heads were selected for the purpose of this study. The selection process was based on the condition that any decrease in accuracy beyond the threshold of 0.005 would render the data unfit for analysis.

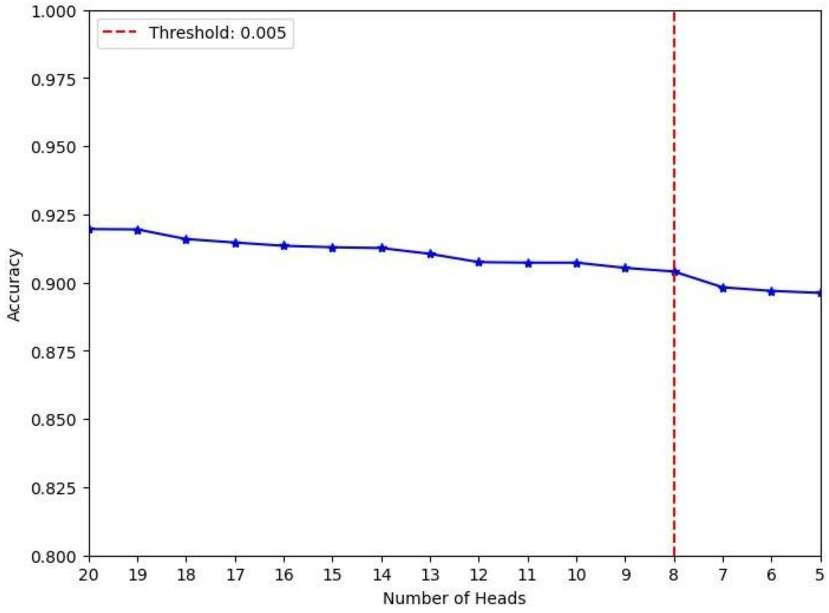


Figure 4.4 Performance metric Accuracy graph w.r.t 40 epochs

The research study demonstrates the impact of removing inefficient heads on the accuracy of the model. The accuracy is evaluated based on the Validation dataset over a period of 40 epochs. The findings are visually represented in Figure 4.4. A Comprehensive Analysis of the Impact of Removing Inefficient Heads on Accuracy Reduction in Validation Datasets Table 4.2 presents a concise comparison of the number of attention heads used in the model and the corresponding accuracy achieved after 20, 30, and 40 epochs of training. The results demonstrate the impact of attention head count on the model's performance during multiple training iterations. Analyzing the accuracy metrics provides insights into the trade-offs between the complexity of the model and its predictive capabilities. This comparison helps identify the optimal attention head configuration that strikes a balance between computational efficiency and predictive accuracy, aiding the selection of the most effective model for story point estimation tasks.

Table 4.2 Performance metric Accuracy table comparison w.r.t 20,30,40 epochs

Number of Heads	Accuracy with 20 epochs	Accuracy with 30 epochs	Accuracy with 40 epochs
20	0.899	0.905	0.920
19	0.898	0.902	0.920
18	0.895	0.901	0.916
17	0.894	0.900	0.915
16	0.891	0.899	0.914
15	0.888	0.896	0.913
14	0.886	0.896	0.913
13	0.883	0.896	0.911
12	0.882	0.889	0.908
11	0.882	0.886	0.907
10	0.879	0.883	0.907
9	0.877	0.883	0.905
8	0.876	0.883	0.904
7	0.870	0.879	0.898
6	0.868	0.879	0.897
5	0.868	0.877	0.896

4.2.2 Performance Measure Based on Mean Absolute Error

The statistical metric employed in this study to assess the accuracy of the proposed GPT2++ model and other baselines is the Mean Absolute Error (MAE). MAE is a commonly used metric in statistical analysis for evaluating the performance of predictive models. The consideration of error magnitude without regard to their signs is a fundamental principle in the field of Mean Absolute Error (MAE) analysis. MAE is a widely used metric in various domains, including statistics, machine learning, and data analysis. This research paper aims to explore the significance of this principle and its implications in the context of error measurement and evaluation. The concept In the context of alternative metrics, it was determined that MdAE, MMRE, and SA were not chosen for further analysis. This decision was based on their inherent limitations in accurately capturing outlier estimates, their tendency to exhibit bias towards underestimation, and their striking resemblance to random guessing. In order to ascertain the statistical significance and effect size of the discrepancy in accuracy between GPT2++ and other baseline models, we utilise a non-parametric adaptation of the ScottKnott ESD test. The present study employs hierarchical clustering as a methodology to categorise median values and detect statistically significant disparities. The Non-Parametric Significance Kernel (NPSK) test is a statistical method utilised in order to minimise the occurrence of false positive results. Unlike other tests, the NPSK test does not depend on assumptions of normality, homogeneous distributions, or a specific minimum sample size. By avoiding these assumptions, the NPSK test provides a more robust and reliable approach to hypothesis testing. The methodology comprises two distinct steps. Firstly, the optimal group divisions are determined by utilising the KruskalChisq statistic, which is based on median values. This statistical measure allows for the identification of the most suitable divisions within the dataset. Secondly, the medians between the established groups are compared to evaluate the extent of differences. This assessment is conducted using the Cliff-JDJ formula, which provides a quantitative measure of the magnitude of disparities between the groups. The ScottKnott ESD (Non-Parametric) analysis is performed utilising the ScottKnott ESD (R) software application (Version 3.0).

Table 4.3 Performance metric Mean Absolute Error table comparison.

Models	Performance Metric (MAE)
GPT2++	0.18
GPT2SP	0.41
GRU-SVM	1.22
BiGRU-SVM	1.48
LSTM-RF	2.1
LSTM-SVM	2.82

The mean absolute error (MAE) in table 4.3 decreases as performance improves. Fundamental techniques for estimating potential situations that may occur during a project. The mean absolute error for each project was determined and recorded also in figure 4.5.

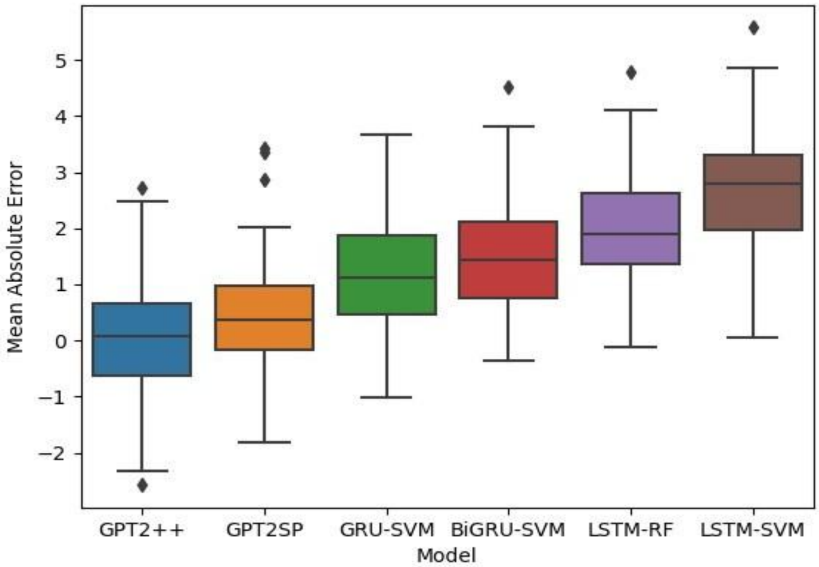


Figure 4.5 Performance matric Mean Absolute Error comparison

4.2.2.1 With-in Project Assessment

An examination of the data gathered throughout the study is done in this section. Models are only used for training and testing inside the framework of a particular project, or "within-project assessment” results are presented in figure 4.6. The datasets linked to each project are arranged chronologically and split into training (60%), validation (20%), and testing (20%) sets to ensure an accurate evaluation of each. The utilization of the same problems between training and testing is prevented by the division of the data into distinct groups for training, validation, and testing. On the testing set, the MAE measure is used to assess GPT2++ models that have been trained on the training set. The 16 datasets are covered by the analysis as part of the internal project evaluation. The MAE is computed for each dataset using the testing data, and the optimal hyper-parameter configurations are identified to minimize the loss value. It's vital to remember that the loss value is obtained from the validation data, whereas the MAE is evaluated on the testing data.

In addition, GPT2++ is compared to five additional methods, GPT2SP[50], LSTM+RF, LSTM+SVM,GRU+SVM, BiGRU+SVM. The outcomes show that GPT2++ functions well. Similar to how Choetkiertikul et al. [50] discussed the use of LSTMs, four machine learning methods—random forest, support vector machine, automatically convert linear models, and linear regression—require an LSTM's vector representation as input. Additionally, Doc2Vec and Bag-of-Words, two additional feature representations, are used to create vector representations. The mean and median story points of the effort estimations are determined, respectively, using the mean and median story points from the training set.

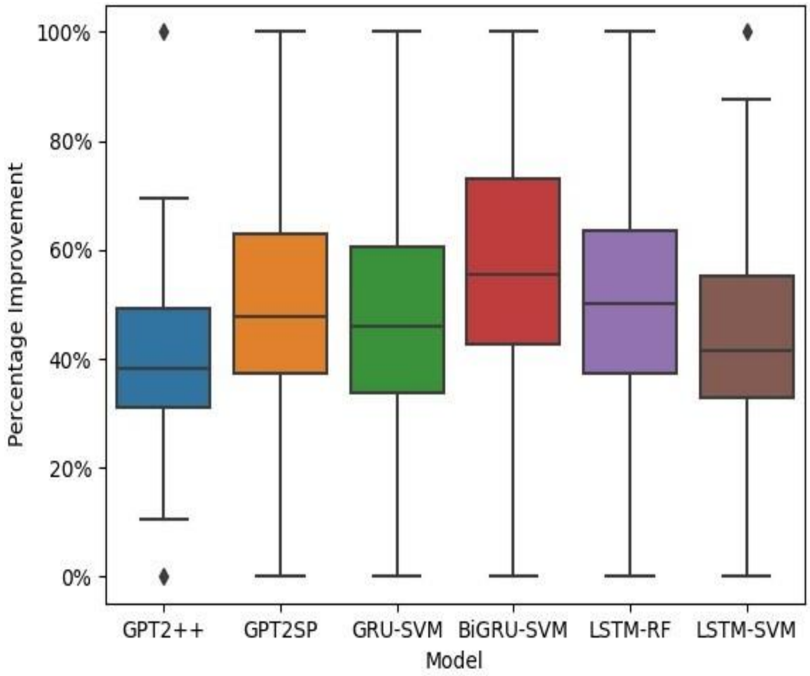


Figure 4.6 Shows the relative improvement in MAE for within-project estimating situations using GPT2++ compared to the baseline comparisons.

4.2.2.2 Cross Project Assessment

It is the process of developing models for one project and then applying those models to another project to see how well they work. We, much as Choetkiertikul et al. [50], pay particular attention to both the linkages between repositories and the links inside individual repositories. The data from one repository is used to train the models, and then the data from another repository is used to assess the models. In figure 4.7 to carry out an assessment inside of a repository, a model must first be trained using data obtained from one repository's project, and then it must be tested with data obtained from another repository's project. Training is performed on the GPT2++models for each project using the training set, and evaluation is performed using the MAE measure on the testing set. Because Choetkiertikul et al. [50] suggests utilizing the same target project for both cross-repository and within-repository assessments, we also follow this recommendation. After that, we compare our GPT2++ to the results of Deep-SE [50] as well as ABE0 (analogy-based estimate). [79], [80], [81], [82]. An estimate in story points is computed by the ABE0 for an issue in the target project by taking the story points of the three problems in the source project that are most like it and average them. Calculating the proportion of the MAE that has improved may be done. In other words, the MAE baseline is set at 100%. baseline MAE ours MAE baseline.

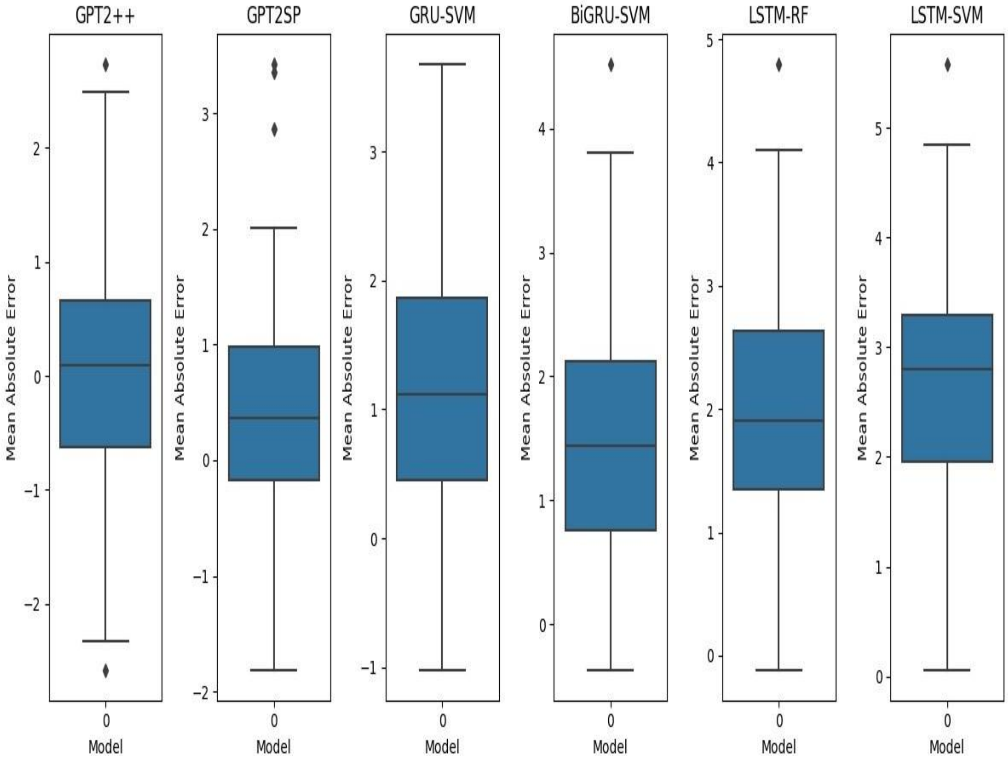


Figure 4.7 MAE for cross-project estimate using GPT2++, Deep-SE, and ABE0

.

The use of GPT-2 language models results in a significant improvement for cross-project estimation scenarios, which demonstrates the advantages of using GPT-2 language models to learn the distributed representations of words in a more general setting. This contrasts with the Deep-SE project-specific pre-trained language models, which were used. Even though DeepSE can generalize its results from one project to another, it is restricted in its capacity to do so because it builds a pre-trained language model for each project to develop a vector representation of each word.

When comparisons are made between evaluations carried out solely inside a repository, GPT2++ performs just as well as Deep-SE and ABE0. It has come to our attention that the median MAE for GPT2++ is 2.4 (Rank-1), whereas that value for Deep-SE is 2.53 (Rank-1), and that value for ABE0 is 2.82 (Rank-2). Even though both GPT2++ and Deep-SE have comparable performance, GPT2++ has been shown to perform better in 62.5% of the tests carried out inside the repository.

4.2.2.3 Performance Based on Sub word Tokenization

To gain a deeper understanding of the topic at hand, it is crucial to thoroughly observe and analyze the mean absolute error of GPT2++ while adjusting its various components as described in figure 4.8. The GPT2++ model comprises two essential components: BPE subword tokenization and the GPT-2 architecture. These components form the foundational elements of GPT2++ and are vital to its functioning. In this study, the focus is on retaining the GPT-2 architecture as the basis for the research while modifying the subword tokenization technique from BPE to either WordPieceSP or SentencePieceSP. This modification aims to enhance the understanding of the impact and significance of subword tokenization in the context of the investigation. By exploring alternative subword tokenization methods, a more comprehensive understanding of the contributions made by different tokenization approaches is sought. The utilization of Word Level tokenization, as seen in Deep-SE, during the transition from the LSTM+RHWN architecture to the GPT-2 architecture allows for a better understanding of the impact introduced by the GPT-2 Transformer. The objective of this study is to examine the mean absolute error (MAE) of five distinct models across sixteen diverse datasets, focusing on the within-project scenario as the experimental framework.

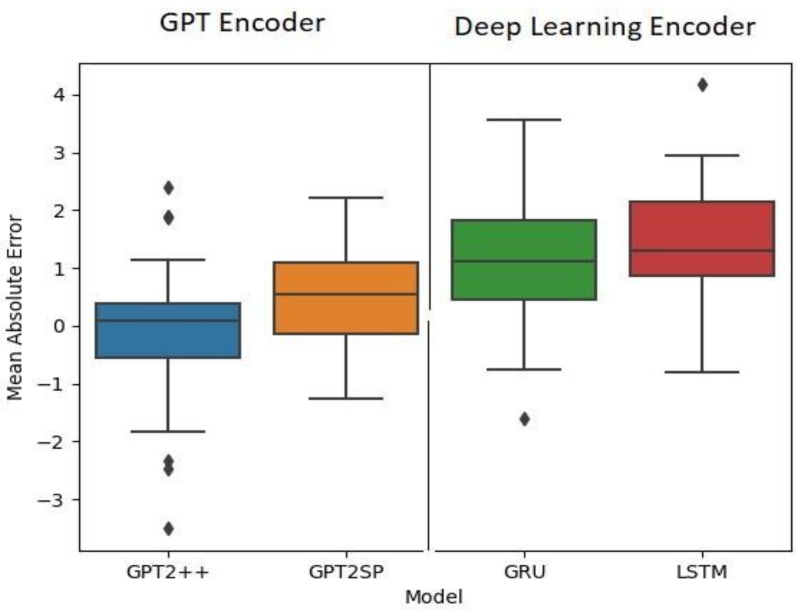


Figure 4.8 shows the effect on GPT2++ model when we alter either the tokenization or the architecture.

However, in the case of GPT2++, BPE remains the optimal subword tokenization strategy. The model is fine-tuned with the hyper-parameter setting that minimizes loss, determined from the validation data, while the mean absolute error (MAE) is calculated using the testing data. The proposed Transformer-based design for GPT2++ significantly reduces the MAE by 67%, indicating a substantial improvement. Comparing different architectures with the same wordlevel tokenization (Word-LevelSP+GPT2 and Word-LevelSP+LSTM+RHWNDeepSE), we observe that Deep-SE experiences a remarkable 6% to 47% improvement in MAE, with a median percentage increase of 34%. This improvement is attributed to the Transformer design employing the masked multi-head self-attention mechanism. Unlike the LSTM unit, which needs to refresh information in the short-term memory cell at each time step, the masked multi-head self-attention approach allows for equal interaction between each word in a sequence, capturing dependencies more accurately and providing richer semantic interpretations. By utilizing a masked mechanism, the models are prohibited from attending to subsequent positions, ensuring their focus remains within the intended context.

When analyzing the results of multiple subword tokenization strategies using the same GPT-2 architecture (BPE+GPT2SP, WordPieceSP+GPT2, and SentencePieceSP+GPT2), minor differences in MAE are observed. This suggests that the GPT-2++ design consistently outperforms the GPT2SP approach, regardless of the subword tokenization strategy employed. The limited impact of tokenization strategies can be attributed to the diverse nature of tasks performed by the Transformer models in subsequent phases. Concerns have been raised by researchers regarding the potential impact of different tokenization strategies on Transformer models used for code generation tasks in software engineering, as it may lead to bugs in the generated code. However, our study demonstrates the robustness and resilience of the Transformer models employed, as the influence of tokenization strategies is found to be limited.

In summary, the GPT2++ model with BPE subword tokenization consistently demonstrates superior performance, resulting in a significant reduction in MAE. The Transformer-based design, incorporating the masked multi-head self-attention mechanism, effectively captures dependencies and semantic interpretations. The findings indicate that the impact of different subword tokenization strategies on the Transformer models used in our analysis is minimal, highlighting their overall resilience.

4.2.3 Performance Based on F1-Score

In this study, we examine the Mean Absolute Error (MAE) distributions of our proposed GPT2++ model and two baseline approaches. To compare the statistical significance of these distributions, we employ a non-parametric variant of the ScottKnott ESD test.The outcomes of this comparative analysis are presented in figure 4.9, which illustrates the performance of our GPT2++ model in relation to the two established methods for cross-project estimation. The investigation involved the determination of the mean absolute error (MAE) for each project under consideration. Furthermore, an evaluation of the performance metrics of our proposed GPT2++

model in comparison to five other state-of-the-art methods is presented in Table 4.4. The experimental findings unequivocally demonstrate that GPT2++ exhibited superior performance compared to other State-of-the-art approaches in terms of accuracy, F-Measure, and Mean Square Error.

Table 4.4 Performance metric F1 Score table comparison.

Models	Performance Metric (F1 Score)
GPT2++	0.87
GPT2SP	0.84
GRU-SVM	0.82
BiGRU-SVM	0.79
LSTM-RF	0.76
LSTM-SVM	0.74

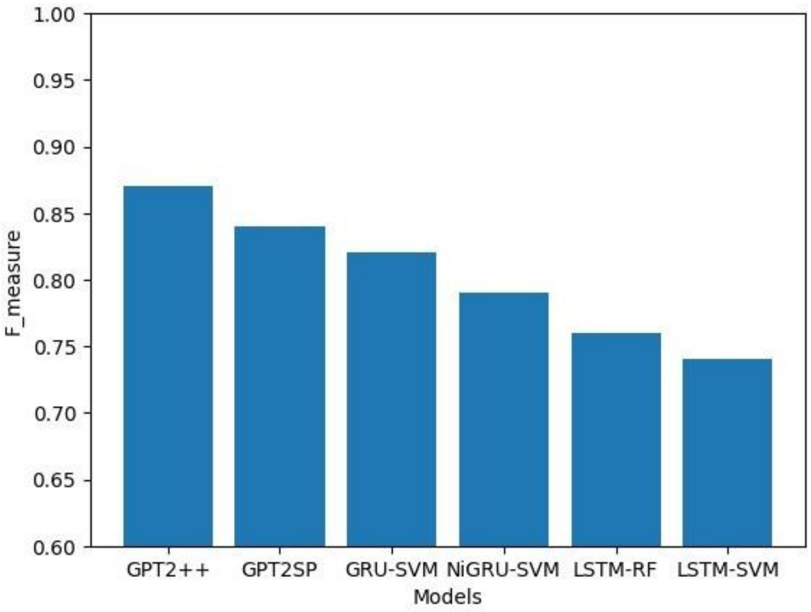
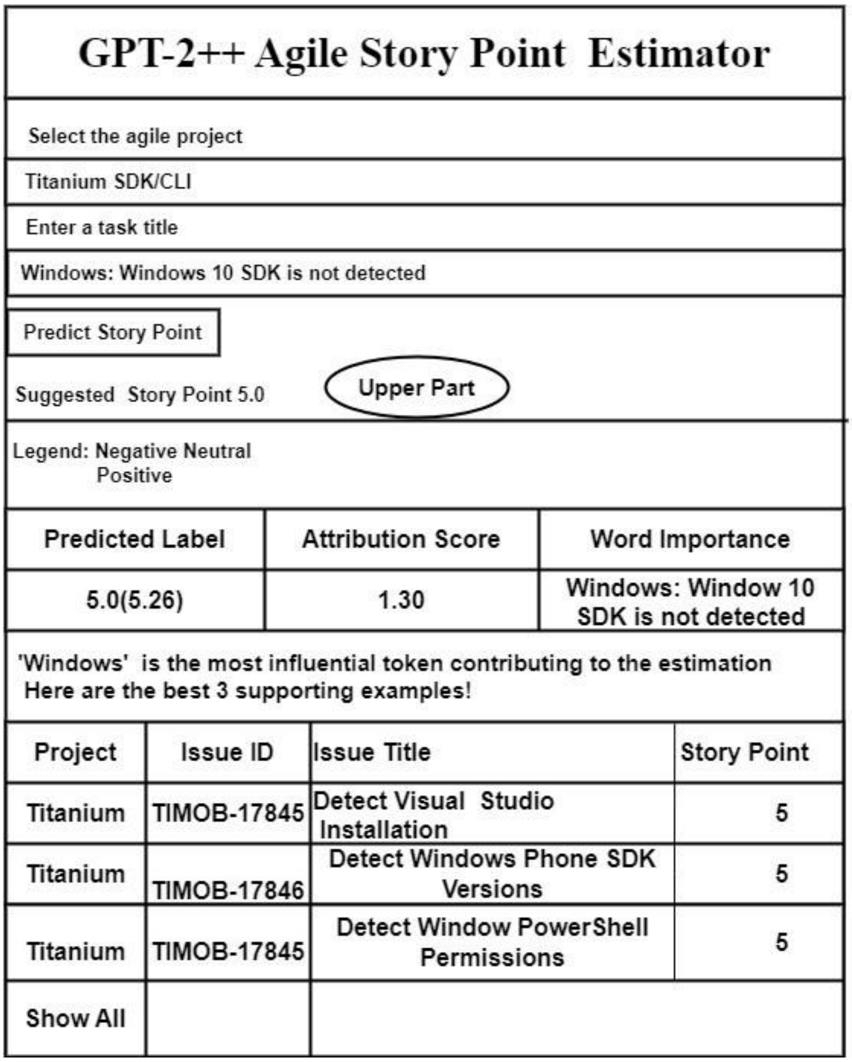


Figure 4.9 Performance metric F1 Score graph comparison.

4.3 GPT2++ Agile Story Point Estimator Tool

In this section, artificial intelligence (AI)-based story point estimation system, which is named GPT2++ presented in figure 4.10. This system has been developed as a web-based tool, equipped with explanations, and supporting examples. The primary objective of the proof-of-concept is to execute a survey study aimed at examining the difficulties associated with story point estimation tasks and to emphasis the importance of providing explanatory support for AI-based story point estimation.



GPT-2++ Agile Story Point Estimator

Select the agile project

Titanium SDK/CLI

Enter a task title

Windows: Windows 10 SDK is not detected

Predict Story Point

Suggested Story Point 5.0

Legend: Negative Neutral
Positive

Predicted Label	Attribution Score	Word Importance
5.0(5.26)	1.30	Windows: Window 10 SDK is not detected

'Windows' is the most influential token contributing to the estimation
Here are the best 3 supporting examples!

Project	Issue ID	Issue Title	Story Point
Titanium	TIMOB-17845	Detect Visual Studio Installation	5
Titanium	TIMOB-17846	Detect Windows Phone SDK Versions	5
Titanium	TIMOB-17845	Detect Window PowerShell Permissions	5
Show All			

Figure 4.10 A snapshot of the proposed GPT2++ tool

To enhance practitioners' comprehension of the estimation process and facilitate the widespread utilization of the GPT2SP model, a web-based story point estimation tool has been developed as a proof-of-concept (refer to Figure 4.10). The tool serves three main purposes, aiming to address a specific issue.

This study focuses on investigating the process of story point estimation in software development projects. The objectives include: **1)** estimating the story point, **2)** identifying the most influential word in the estimation process, and **3)** providing supporting examples from the training set of the project under consideration. The **first objective** is to accurately estimate the story point, which is a commonly used metric in agile software development for quantifying the effort required to complete a user story or task. Accurate story point estimation enables effective project planning and resource allocation. Various techniques and methodologies employed by software development teams for story point estimation will be explored. The **second objective** is to highlight the key word or phrase that significantly contributes to story point estimation. By identifying these influential words, insights can be gained into the factors that influence estimation and potentially improve accuracy. This analysis will involve examining a dataset of user stories. The supporting examples used in this study are selected based on their inclusion of the most significant keyword and having the same story point as the target issue.

The research paper focuses on the utilization of two concepts of Explainable Artificial Intelligence (AI) in the development of the GPT2++tool. The **third objective** concepts are feature-based explanations and example-based explanations. Incorporating these concepts enhances the interpretability and transparency of the AI system. Feature-based explanations assist practitioners in understanding the key words that significantly influence story point estimation for a given issue. These explanations provide valuable insights into the estimation process factors. By identifying the most important words, practitioners can enhance their understanding and make informed decisions. On the other hand, example-based explanations extend the casebased reasoning paradigm, where optimal supporting examples are searched based on identical words and story points within the same project.

4.3.1 Example Usage of Tool GPT2++

Application of the GPT2SP Tool. An issue (TIMOB-20252) from the Titanium project is used as an example to demonstrate the use of the GPT2++ tool. The title of the problem is "Windows: Windows 10 SDK is not detected." When you enter this title into the GPT2++tool, the model predicts a story point of 5.0. Based on the actual ground-truth, it is discovered that the story point estimation for this issue is correct. The GPT2++ utility offers two main explanations for this problem. To begin, it determines the most important word that contributed to the estimation of tale points, which in this case is "Windows." This suggests that the presence of the phrase "Windows" was important in identifying the story point. Furthermore, based on the most important word, the GPT2++ tool provides the top three supporting instances. TIMOB-178452, TIMOB-178463, and TIMOB-178474 are three examples. These instances are related by the word "Windows" and have the same story elements as the target issue. This finding implies that topics with comparable story elements frequently have similar keywords. As a result, the GPT2++ tool may help Agile teams achieve consistency in story point prediction using historical data.

Overall, this example exhibits the GPT2++ tool's actual use, demonstrating its capacity to estimate story points accurately while also providing valuable explanations and accompanying examples to aid in the estimation process.

4.4 Summary

Mean Absolute Error (MAE) is applied in both the internal and external experiments of this study, and it is also utilized by Choetkiertikul et al. to evaluate DeepSE. This research was carried out in the United Kingdom. The mean absolute error (MAE) is a statistical measure that examines how much various predictions disagree, ignoring the directions in which the errors are made. Other metrics such as MdAE, MMRE, and SA are not preferred since they are unable to capture outlier estimates, they have a bias toward underestimating, and they are comparable to random guessing. Calculating the significance of the accuracy gap between GPT2++ and other baselines is accomplished with the use of the non-parametric ScottKnott ESD test. This test ranks treatments according to mean values and ensures that there are major variations between groups. When conducting internal evaluations, datasets are often segmented into training, validation, and testing sets to eliminate temporal validation bias. When training GPT2++ models, the training set is used; after that, the models are evaluated using the MAE metric on the testing set. During cross-project evaluations, which take place when models that were trained on one repository are evaluated on another, GPT2++ is compared to Deep-SE and ABE0. GPT2++ performs much better than the baselines and is statistically preferable because of its lower mean absolute error (MAE) value. Ablation study is used to investigate how the performance of GPT2++ is affected by the interactions between BPE sub word tokenization and the GPT-2 architecture. The research results show that GPT2++, which makes use of the Transformer design, significantly improves Deep-SE's MAE, but the other sub word tokenization techniques have just a little impact on MAE. This is because GPT2++ uses the Transformer architecture.

Chapter 5

Conclusion and Future Work

Chapter 5

Conclusion and Future Work

5.1 Overview of Research

The purpose of this project is to find solutions to the problems that software development teams have when attempting to provide realistic estimates of the complexity and work necessary to finish user stories. Although algorithms already exist, there is a lack of understanding of the underlying context of user wants, which leads to outputs that are less than optimum and missed deadlines. The methodologies of machine learning and deep learning both have drawbacks, the most notable of which are their high time complexity and low accuracy.

Pre-trained transformers, GPT models, have showed potential in improving story-point estimation as a means of overcoming the issues described above. However, within the framework of the GPT-2++ model, there is a possibility that some attention heads may not effectively contribute to the process of estimating story points, which will result in findings that are not as good as they might be. The purpose of this study is to determine which attention heads are not productive and then get rid of them to get a more accurate estimate of the number of story points.

The purpose of this study is to make the process of computing story points more precise, predictable, and time-efficient by getting rid of attention heads that aren't doing their job. Among the anticipated results is an improvement in both the predictability of and the efficiency with which one can estimate the amount of work necessary for user stories. This increase in estimating accuracy and efficiency would be beneficial to project management since it would enable improved planning and resource allocation.

Collecting a varied dataset of user stories and the story points that correlate to those stories is an integral part of the study technique.

Following the completion of preprocessing, which involves cleaning and transforming the data, the GPT-2++ model is fine-tuned by making use of the dataset. A study of the attention heads is carried out to determine which attention heads are ineffective. Afterward, the ineffective attention heads are removed by making the necessary adjustments.

The performance of the updated GPT-2++ model is evaluated based on the given evaluation metrics, and the results are compared to baseline models and current estimating techniques. Experimentation and validation are components of the study, both of which are used to establish the dependability and generalizability of the findings.

By enhancing attention processes in pre-trained transformers, the results of this study hope to contribute to the area of software estimation. The projected results include a more precise and efficient assessment of story points, which will lead to improved software project planning, monitoring, and execution in the real world.

In conclusion, the purpose of this study is to improve story-point estimate by getting rid of attention heads in the GPT-2++ model that aren't very successful. Data collection, preprocessing, model training, attention head analysis, elimination, assessment, and validation are all components of this technique. The anticipated advantages include increased accuracy and efficiency in estimating costs, which would eventually lead to more dependable project management and efficient use of available resources.

5.2 Summary of Research Contributions

The discipline of software estimating, and story-point estimation in particular, benefits from several important advances made by this study.

- **Identification of Challenges:** The findings of this study give a full knowledge of the issues that software development teams confront when attempting to correctly estimate the level of complexity and labor necessary to execute user stories. It brings to light the restrictions imposed by currently available algorithms as well as the challenges related with machine learning and deep learning strategies.
- **Integration of Pre-trained Transformers:** The project investigates the possibility of improving the accuracy of story-point estimate by exploiting sophisticated natural language processing capabilities.

This is done by introducing pre-trained transformers, notably the GPT-2++ model, into the estimation process.

- **Attention Head Analysis:** The study presents an original method for assessing attention heads within the context of the GPT-2++ model. It does this by analyzing the attention patterns and weights, which allows it to determine which attention heads do not successfully contribute to an accurate evaluation of the story points.
- **Attention Head Elimination:** Using the analysis as a foundation, the study proposes a strategy for removing inefficient attention heads from the GPT-2++ model. To improve estimating precision and performance, this procedure entails adjusting the weights of the remaining attention heads or altering the processes that control attention.
- **Improved Estimation Accuracy and Efficiency:** According to the findings of the study, the updated GPT-2++ model can achieve improved estimate accuracy and efficiency in the process of computing story points by getting rid of attention heads that are inefficient. This enhancement may have a substantial influence on the planning, monitoring, and resource allocation of the project.
- **Practical Application and Real-World Impact:** The results of this study have ramifications that may be used in practice for software development teams working in the "real world." Teams can more effectively plan, monitor progress, and execute projects, which leads

to fewer delays and better project results when they have the capacity to estimate story points more correctly.

- **Advancement of Research in Software Estimation:** This study contributes to the more general topic of software estimation by looking at many novel techniques that might improve attention processes in pre-trained transformers. It provides insights into enhancing the estimate accuracy of complicated tasks via the integration of modern approaches for processing natural languages.

This study adds to the knowledge and enhancement of story-point estimation by addressing issues in reliably estimating the effort necessary to complete user stories. In summary, the research addresses challenges in accurately estimating the amount of work that is required to accomplish user stories. To improve both the accuracy and the efficiency of the estimating process, the approach makes use of pre-trained transformers, evaluates attention heads, and gets rid of the inefficient ones. The relevance of this study is further validated by its practical ramifications as well as the developments that have been achieved in software estimation.

5.3 Conclusion of the Research

In conclusion, the evaluation of the difficulty and effort involved in user stories is critical to the production of productive software development cycles. However, the currently available algorithms for story-point task evaluation usually fail to comprehend the contextual complexities of user requirements, which results in delays and missed deadlines. Both machine learning and deep learning have been hampered by issues such as high time complexity and poor accuracy, despite the promise that both hold. The development of pre-trained transformers, particularly GPT, has, on the other hand, significantly contributed to a reduction in the severity of these issues. However, there is a possibility that some focus areas in GPT models may not successfully contribute to the estimation of story points, which may result in results that are less than desirable. This study intends to enhance story point estimation by identifying inefficient focus areas inside the GPT-2 model and then removing them from the model entirely. The objective is to simplify the process of producing an accurate estimate of the amount of time and effort needed to accomplish a certain number of story points. By addressing these issues, we will hopefully be able to make software development teams more productive as a whole and make it easier to evaluate the degree of difficulty associated with user stories.

5.4 Future Work

The results that were provided in this abstract have paved the path for further study into enhancing the story point estimation process used in the software sector. The following are examples of potential subjects for further research:

The present analysis focuses on recognizing and eliminating wasted attention heads within the GPT-2 model however, more research may be undertaken to examine methods of improving attention mechanisms that are custom-tailored for story-point estimation. In the meantime, this examination is centered on the GPT-2 model. One strategy would be to investigate a variety of attention structures, while another would be to devise innovative approaches to increase the significance of the attention heads and their contribution to the estimating task. Both strategies would be useful.

An in-depth understanding of the context around user needs is necessary for an accurate evaluation of the story points. Future work may study methods to combine domain-specific information, industry-specific ontologies, or other data sources to get a better understanding of the context of user stories. This will allow for a more comprehensive understanding of the user stories themselves. Because of this, it may be necessary to make use of unique pre-training methods that include the use of datasets that are relevant to a certain domain or the inclusion of external knowledge graphs.

It is feasible to further improve the software development lifecycle by integrating the updated story point estimate method with extant project management tools and frameworks. This may be done to achieve the goal of further optimizing the software development lifecycle. In the future, we will be able to provide connectors and plugins for well-known project management software to facilitate the process of incorporating the estimating procedure into the program. The incorporation of accurate story-point predictions into the processes of planning and monitoring in such a manner would be of tremendous use to teams.

Extensive benchmarking and comparative study of different story-point estimate approaches, including the proposed enhanced GPT-based method, will give helpful insights into their relative strengths and constraints. These insights may be used to improve the suggested method. To do this, we may evaluate the effectiveness of various algorithms by using a standard set of evaluation criteria and data sets. The findings of such an investigation would be helpful in establishing which approaches are the most efficient, as well as which approaches to estimating are the most appropriate for a wide range of circumstances.

Before the suggested innovations can be implemented, they need to first be tested and their performance assessed in real software development projects. It would be beneficial for future study to explore the possibility of putting the enhanced story point estimate process into practice by collaborating with key industry stakeholders. This would make it feasible to obtain genuine data, assess performance in real-world scenarios, and request user input for future enhancements. All these things would be achievable thanks to this.

By studying these potential work approaches, researchers and practitioners may continue to enhance story-point estimates in software development. This will result in more exact, efficient, and trustworthy estimating techniques, which will eventually contribute to the success of software projects.

5.5 Summary

By addressing the problems that now exist, the purpose of this study is to enhance the accuracy and productivity of story-point estimate in software development teams. We emphasize the limits of the algorithms and techniques to machine learning that are currently in use, and we investigate the possibility of pre-trained transformers, more especially the GPT-2++ model. To improve the accuracy of the estimate process, the study is focused on locating and removing inefficient attention heads from inside the GPT-2++ model. The redesigned model delivers higher estimating accuracy and efficiency because of the removal of these ineffective attention heads. This, in turn, leads to enhanced project planning, monitoring, and resource allocation. The study provides several key advances, including the identification of difficulties, the integration of pre-trained transformers, the introduction of attention head analysis and deletion, and the demonstration of practical application and real-world effect. The results provide a contribution to the progression of research in software estimate and have consequences for the improvement of project outcomes in the industry of software development.

In the future, additional research can be carried out to optimize attention mechanisms that are tailored for story-point estimation, investigate ways to incorporate domain-specific information, integrate the revised estimation process with project management tools, perform benchmarking and comparative studies, and validate the proposed enhancements in actual software development projects. These prospective directions intend to enhance the accuracy, efficacy, and dependability of the techniques used for estimating story points, which will eventually contribute to the success of software projects.

References

[1] Dhar, V., Jarke, M. and Laartz, J. (2014). Big Data. WIRTSCHAFTSINFORMATIK, 56(5), pp.277–279. doi:10.1007/s11576-014-0428-0.

[2] Flyvbjerg, B. and Budzier, A. (2011). Why Your IT Project May Be Riskier than You Think. SSRN Electronic Journal. doi:10.2139/ssrn.2229735.

[3] Jørgensen, M. (2016). Unit effects in software project effort estimation: Workhours gives lower effort estimates than workdays. Journal of Systems and Software, 117, pp.274–281. doi: 10.1016/j.jss.2016.03.048.

[4] Wieczorek, Ł. and Ignaciuk, P. (2018). Continuous Genetic Algorithms as Intelligent Assistance for Resource Distribution in Logistic Systems. Data, 3(4), p.68. doi:10.3390/data3040068.

[5] Misirli, A.T., Caglayan, B., Bener, A. and Turhan, B. (2013). A Retrospective Study of Software Analytics Projects: In-

Depth Interviews with Practitioners. IEEE Software, 30(5), pp.54–61. doi:10.1109/ms.2013.93.

[6] Jorgensen, M. (2014). What We Do and Don’t Know about Software Development Effort Estimation. IEEE Software, 31(2), pp.37–40. doi:10.1109/ms.2014.49.

[7] McConnell, S. (1998). The art, science, and engineering of software development. IEEE Software, 15(1), pp.120, 118–119. doi:10.1109/52.646892.

[8] Jorgensen, M. and Shepperd, M. (2007). A Systematic Review of Software Development Cost Estimation Studies. IEEE

Transactions on Software Engineering, 33(1), pp.33–53. doi:10.1109/tse.2007.256943.

[9] Boehm, B., Egyed, A., Port, D., Shah, A., Kwan, J. and Madachy, R. (1998). Annals of Software Engineering, 6(1/4), pp.295–

321. doi:10.1023/a:1018988827405.

[10] Sentas, P., Angelis, L. and Stamelos, I. (2007). A statistical framework for analyzing the duration of software projects. Empirical Software Engineering, 13(2), pp.147–184. doi:10.1007/s10664-007-9051-7.

[11] Sentas, P., Angelis, L., Stamelos, I. and Bleris, G. (2005). Software productivity and effort prediction with ordinal regression. Information and Software Technology, 47(1), pp.17–29. doi: 10.1016/j.infsof.2004.05.001.

[12] Cervone, H.F. (2011). Understanding agile project management methods using Scrum. OCLC Systems & Services: International digital library perspectives, 27(1), pp.18–22. doi:10.1108/10650751111106528.

[13] III, W.G. (1987).: Generations: A Chinese Family. Richard Gordon, Carma Hinton, Kathy Kline. American Anthropologist, 89(1), pp.255–256. doi:10.1525/aa.1987.89.1.02a01150.

[14] Choetkiertikul, M., Dam, H.K., Tran, T., Ghose, A. and Grundy, J. (2018). Predicting Delivery Capability in Iterative Software Development. IEEE Transactions on Software Engineering, 44(6), pp.551–573. doi:10.1109/tse.2017.2693989. [15] Fu, M. and Tantithamthavorn, C. (2022). GPT2SP: A Transformer-Based Agile Story Point Estimation Approach. IEEE Transactions on Software Engineering, pp.1–1. doi:10.1109/tse.2022.3158252.

[16] Vestergaard, E.T. and Jorgensen, J.O.L. (2006). Role of ghrelin in growth hormone-deficient patients. Expert Review of Endocrinology & Metabolism, 1(3), pp.343–351. doi:10.1586/17446651.1.3.343.

[17] Jorgensen, M. and Gruschke, T.M. (2009). The Impact of Lessons-Learned Sessions on Effort Estimation and Uncertainty Assessments. IEEE Transactions on Software Engineering, 35(3), pp.368–383. doi:10.1109/tse.2009.2.

[18] Boehm, B., Egyed, A., Port, D., Shah, A., Kwan, J. and Madachy, R. (1998). Annals of Software Engineering, 6(1/4), pp.295–321. doi:10.1023/a:1018988827405.

[19] Sentas, P., Angelis, L. and Stamelos, I. (2007). A statistical framework for analyzing the duration of software projects. Empirical Software Engineering, 13(2), pp.147–184. doi:10.1007/s10664-007-9051-7.

[20] Niranjan, R., Kanmani, D. and Kumar, S. (2022). Design and fabrication of multi cutting hack saw

machine. International journal of health sciences, pp.10294–10297. doi: 10.53730/ijhs.v6ns3.9423.

[21] Panda, S.K., Satapathy, S.P., Panda, P.C., Lakra, K., Karir, S. and Panda, J.N. (2021). Determinants of Inpatient Satisfaction on Hospital Services in a Government Tertiary Care Center. Healthline, 12(2), pp.5–12. doi:10.51957/healthline_202_2021.

[22] Bibi, S., Stamelos, I. and Angelis, L. (2008). Combining probabilistic models for explanatory productivity estimation. Information and Software Technology, 50(7-8), pp.656–669. doi: 10.1016/j.infsof.2007.06.004.

[23] Shepperd, M. and Schofield, C. (1997). Estimating software project effort using analogies. IEEE Transactions on Software Engineering, 23(11), pp.736–743. doi:10.1109/32.637387.

[24] Karoulis, A., Stamelos, I.G., Angelis, L. and Pombortsis, A.S. (2005). Formally Assessing an Instructional Tool: A Controlled Experiment in Software Engineering. IEEE Transactions on Education, 48(1), pp.133–139.
doi:10.1109/te.2004.837047.

[25] Collopy, F. (2007). Difficulty and complexity as factors in software effort estimation. International Journal of Forecasting, 23(3), pp.469–471. doi: 10.1016/j.ijforecast.2007.05.011.

[26] Tawosi, V., Sarro, F., Petrozziello, A. and Harman, M. (2022). Multi-Objective Software Effort Estimation: A Replication Study. IEEE Transactions on Software Engineering, 48(8), pp.3185–3205. doi:10.1109/tse.2021.3083360. [27] Kocaguneli, E., Menzies, T., and Keung, J.W. (2012). On the Value of Ensemble Effort Estimation. IEEE Transactions on Software Engineering, 38(6), pp.1403–1416. doi:10.1109/tse.2011.111.

[28] Valerdi, R. (2011). 10.4.2 Convergence of Expert Opinion via the Wideband Delphi Method: An Application in Cost Estimation Models. INCOSE International Symposium, 21(1), pp.1246–1259. doi:10.1002/j.2334-5837. 2011.tb01282. x.

[29] Chulani, S., Boehm, B. and Steece, B. (1999). Bayesian analysis of empirical software engineering cost models. IEEE Transactions on Software Engineering, 25(4), pp.573–583. doi:10.1109/32.799958

[30] Moreno, A. (2006). Review of 'Agile Estimating and Planning by Mike Cohn,' Prentice Hall PTR, 2005, $44.99, ISBN: 0131479415. Queue, 4(5), p.59. doi:10.1145/1142031.1142049.

[31] Concas, G., Marchesi, M., Murgia, A., Tonelli, R. and Turnu, I. (2011). On the Distribution of Bugs in the Eclipse

System. IEEE Transactions on Software Engineering, 37(6), pp.872–877. doi:10.1109/tse.2011.54.

[32] Djouab, R., Abran, A. and Seffah, A. (2014). An ASPIRE-based method for quality requirements identification from business goals. Requirements Engineering, 21(1), pp.87–106. doi:10.1007/s00766-014-0211-1.

[33] Pedrycz, W., Succi, G., Sillitoe, A. and Algazi, J. (2015). Data description: A general framework of information granules. Knowledge-Based Systems, 80, pp.98–108. do: 10.1016/j.knosys.2014.12.030.

[34] Neil, M., Tailor, M., Marquez, D., Fenton, N. and Hearty, P. (2008). Modelling dependable systems using hybrid Bayesian networks. Reliability Engineering & System Safety, 93(7), pp.933–939. doe: 10.1016/j.ress.2007.03.009.

[35] Silva, L., Almeida, H., Perkusich, A. and Perkusich, M. (2015). A Model-Based Approach to Support Validation of Medical Cyber-Physical Systems. Sensors, 15(11), pp.27625–27670. doi:10.3390/s151127625.

[36] Pinzger, M., Giger, E., and Gall, H.C. (2021). Comparing fine-grained source code changes and code churn for bug prediction - A retrospective. ACM SIGSOFT Software Engineering Notes, 46(3), pp.21–23. doi:10.1145/3468744.3468751. [37] Fackler, M. (2021). Panjer class revisited: one formula for the distributions of the Panjer (a, b, n) class. SSRN Electronic Journal. doi:10.2139/ssrn.3813246.

[38] Bhattacharya, P., Neamtiu, I. and Shelton, C.R. (2012). Automated, highly accurate, bug assignment using machine learning and tossing graphs. Journal of Systems and Software, 85(10), pp.2275–2292. doi: 10.1016/j.jss.2012.04.053.

[39] Hooimeijer, P. and Weimer, W. (2012). StrSolve: solving string constraints lazily. Automated Software Engineering, 19(4), pp.531–559. doi:10.1007/s10515-012-0111-x.

[40] Choetkiertikul, M., Dam, H.K., Tran, T., Ghose, A. and Grundy, J. (2018). Predicting Delivery Capability in Iterative Software Development. IEEE Transactions on Software Engineering, 44(6), pp.551–573. doi:10.1109/tse.2017.2693989. [41] Linares-Vásquez, M., Vendome, C., Tufano, M. and Poshyvanyk, D. (2017). How developers micro-optimize Android apps. Journal of Systems and Software, 130, pp.1–23. doi: 10.1016/j.jss.2017.04.018.

[42] Dam, H.K., Tran, T. and Pham, T. (2016). A deep language model for software code. arXiv:1608.02715 [cs, stat].

[online] Available at: http://arxiv.org/abs/1608.02715 [Accessed 5 Jan. 2023].

[43] X. Gu, H. Zhang, D. Zhang, and S. Kim, “Deep API learning,” in Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng., 2016, pp. 631–642. [Online]. Available: http://doi.acm.org/10.1145/ 2950290.2950334

[44] Gupta, R., Pal, S., Kanade, A. and Shevade, S. (2017). DeepFix: Fixing Common C Language Errors by Deep Learning. Proceedings of the AAAI Conference on Artificial Intelligence, 31(1). doi:10.1609/aaai. v31i1.10742.

[45] Hinton, G.E. (2006). Reducing the Dimensionality of Data with Neural Networks. Science, [online] 313(5786), pp.504–507. doi:10.1126/science.1127647.

[46] Atlassian, Jira — Issue & Project Tracking Software — Atlassian. [Online]. Available: https: //www.atlassian.com/software/jira.

[47] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, “Agile software development methods: Review and analysis,” 2017, arXiv: 1709.08439.

[48] M. Usman, E. Mendes, and J. Borstler, “Effort estimation in agile€ software development: A survey on the state of the practice,” in Proc. 19th Int. Conf. Eval. Assessment Softw. Eng., 2015, pp. 1–10.

[49] M. Usman, E. Mendes, F. Weidt, and R. Britto, “Effort estimation in agile software development: A systematic literature review,” in Proc. 10th Int. Conf. Predictive Models Softw. Eng., 2014, pp. 82–91.

[50] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, “A deep learning model for estimating story points,” IEEE Trans. Softw. Eng., vol. 45, no. 7, pp. 637–656, Jul. 2019.

[51] M. Jørgensen, “A review of studies on expert estimation of software development effort,” *Journal of Systems and*

Software, Vol. 70, No. 1–2, 2004, pp. 37–60.

[52] B. Boehm, *Software cost estimation with COCOMO II*. New Jersey: Prentice-Hall, 2000.

[53] P. Sentas, L. Angelis, and I. Stamelos, "Multinomial logistic regression applied on software productivity prediction," in *9th Panhellenic Conference in Informatics*, 2003, pp. 1–12.

[54] S. Kanmani, J. Kathiravan, S.S. Kumar, and M. Shanmugam, "Neural network-based effort estimation using class points for OO systems," in *International Conference on Computing: Theory and Applications (ICCTA'07)*. IEEE, 2007, pp. 261– 266.

[55] A. Panda, S.M. Satapathy, and S.K. Rath, "Empirical validation of neural network models for agile software effort estimation based on story points," *Procedia Computer Science*, Vol. 57, 2015, pp. 772–781.

[56] S. Kanmani, J. Kathiravan, S.S. Kumar, and M. Shanmugam, "Class point based effort estimation of oo systems using fuzzy subtractive clustering and artificial neural networks," in *Proceedings of the 1st India Software Engineering Conference*, 2008, pp. 141–142.

[57] S. Bibi, I. Stamelos, and L. Angelis, "Software cost prediction with predefined interval estimates," in *Proceedings of Software Measurement European Forum*, Vol. 4, 2004, pp. 237–246.

[58] L. Angelis and I. Stamelos, "A simulation tool for efficient analogy-based cost estimation," *Empirical Software Engineering*, Vol. 5, No. 1, 2000, pp. 35–68.

[59] F. Sarro, A. Petrozziello, and M. Harman, "Multi-objective software effort estimation," in *38th International*

Conference on Software Engineering (ICSE). IEEE, 2016, pp. 619–630.

[60] M. Cohn, *Agile estimating, and planning*. Pearson Education, 2005.

[61] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, "Estimating story points from issue reports," in *Proceedings of the the 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, 2016, pp. 1–10.

[62] C. Commeyne, A. Abran, and R. Djouab, “Effort estimation with story points and cosmic function points – An industry case study,” *Software Measurement News*, Vol. 21, No. 1, 2016, pp. 25–36.

[63] G. Poels, “Definition and validation of a COSMIC-FFP functional size measure for objectoriented systems,” in *Proc. 7th Int. ECOOP Workshop Quantitative Approaches OO Software Eng. Darmstadt*, 2003.

[64] P. Abrahamsson, R. Moser, W. Pedrycz, A. Sillitti, and G. Succi, “Effort prediction in iterative software development processes – Incremental versus global prediction models,” in *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*. IEEE, 2007, pp. 344–353.

[65] P. Hearty, N. Fenton, D. Marquez, and M. Neil, “Predicting project velocity in XP using a learning dynamic 92ayesian network model,” *IEEE Transactions on Software Engineering*, Vol. 35, No. 1, 2008, pp. 124–137.

[66] M. Perkusich, H.O. De Almeida, and A. Perkusich, “A model to detect problems on scrum-based software development projects,” in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 1037–1042.

[67] E. Giger, M. Pinzger, and H. Gall, “Predicting the fix time of bugs,” in *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, 2010, pp. 52–56.

[68] L.D. Panjer, “Predicting eclipse bug lifetimes,” in *Fourth International Workshop on Mining Software Repositories (MSR’07: ICSE Workshops 2007)*. IEEE, 2007, pp. 29–29.

[69] P. Bhattacharya and I. Neamtiu, “Bug-fix time prediction models: Can we do better?” in *Proceedings of the 8th Working Conference on Mining Software Repositories*, 2011, pp. 207–210.

[70] P. Hooimeijer and W. Weimer, “Modeling bug report quality,” in *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering*, 2007, pp. 34–43.

[71] E.M.D.B. Fávero, D. Casanova, and A.R. Pimentel, “SE3M: A model for software effort estimation using pre-trained embedding models,” *Information and Software Technology*, Vol. 147, 2022, p. 106886.

[72] P. Liu, Y. Liu, X. Hou, Q. Li, and Z. Zhu, “A text clustering algorithm based on find of density peaks,” in *7th International Conference on Information Technology in Medicine and Education (ITME)*. IEEE, 2015, pp. 348–352.

[73] J. Pennington, R. Socher, and C.D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[74] W. Guohua and G. Yutian, “Using density peaks sentence clustering for update summary generation,” in *Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2016, pp. 1–5.

[75] F. Sarro, A. Petrozziello, and M. Harman, “Multi-objective software effort estimation,” in Proc. IEEE/ACM 38th Int. Conf. Softw. Eng., 2016, pp. 619–630.

[76] C. Tantithamthavorn, A. E. Hassan, and K. Matsumoto, “The impact of class rebalancing techniques on the performance and interpretation of defect prediction models,” IEEE Trans. Softw. Eng., vol. 46, no. 11, pp. 1200–1219, Nov. 2020.

[77] M. Shepperd and S. MacDonell, “Evaluating prediction systems in software project estimation,” Inf. Softw. Technol., vol. 54, no. 8, pp. 820–827, 2012.

[78] J. W. Keung, B. A. Kitchenham, and D. R. Jeffery, “Analogy-X: Providing statistical inference to analogy-based software cost estimation,” IEEE Trans. Softw. Eng., vol. 34, no. 4, pp. 471–484, Jul./Aug. 2008.

[79] E. Kocaguneli, T. Menzies, A. Bener, and J. W. Keung, “Exploiting the essential assumptions of analogy-based effort estimation,” IEEE Trans. Softw. Eng., vol. 38, no. 2, pp. 425–438, Mar./Apr. 2012.

[80] E. Kocaguneli, T. Menzies, and E. Mendes, “Transfer learning in effort estimation,” Empir. Softw. Eng., vol. 20, no.

3, pp. 813–843, 2015.

[81] Y.-F. Li, M. Xie, and T. N. Goh, “A study of project selection and feature weighting for analogy-based software cost estimation,” J. Syst. Softw., vol. 82, no. 2, pp. 241–252, 2009.

[82] Y. Ding, B. Ray, P. Devanbu, and V. J. Hellendoorn, “Patching as translation: The data and the metaphor,” in Proc. 35th IEEE/ACM Int. Conf. Automated Softw. Eng., 2020, pp. 275–286

[83] N. Jiang, T. Lutellier, and L. Tan, “CURE: Code-aware neural machine translation for automatic program repair,” in Proc. Int. Conf. Softw. Eng., 2021, pp. 1161–1173.

[84] R.-M. Karampatsis, H. Babii, R. Robbes, C. Sutton, and A. Janes, “Big code! = Big vocabulary: Open-vocabulary models for source code,” in Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng., 2020, pp. 1073–1085.

[85] . Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, “Estimating story points from issue reports,” in Proc. 12th Int. Conf. Predictive Models Data Anal. Softw. Eng., 2016, pp. 1–10.

[86] O. Liskin, R. Pham, S. Kiesling, and K. Schneider. Why we need a granularity concept for user stories. In Agile Processes in Software Engineering and Extreme Programming, pages 110–125. Springer, 2014.

[87] K. Molokken-Ostvold and K. M. Furulund. The relationship between customer collaboration and software project overruns. In Agile Conference (AGILE), 2007, pages 72–83. IEEE, 2007.

[88] K. Molokken-Ostvold and M. Jorgensen. A comparison of software project overruns-flexible versus sequential development models. Software Engineering, IEEE Transactions on, 31(9):754–766, 2005.

[89] K. Moløkken-Østvold, M. Jørgensen, S. S. Tanilkan, H. Gallis, A. C. Lien, and S. Hove. A survey on software estimation in the norwegian industry. In Software Metrics, 2004. Proceedings. 10th International Symposium on, pages 208–219.
IEEE, 2004

[90] M. Usman, E. Mendes, F. Weidt, and R. Britto. Effort estimation in agile software development: A systematic literature review. In Proceedings of the 2014 International Conference on Predictive Models in Software Engineering, pages 82–91. ACM, 2014.

[91] R. T. Hughes. Expert judgement as an estimating method. Information and Software Technology, 38(2):67–75, 1996.

[92] T. K. Abdel-Hamid. Investigating the cost/schedule trade-off in software development. Software, IEEE, 7(1):97–105, 1990

[93] T. DeMarco. Controlling software projects: Management, measurement, and estimates. Prentice Hall PTR, 1986.

[94] W. AbdelMoez, M. Kholief, and F. M. Elsalmy. Improving bug fix-time prediction model by filtering out outliers. In Proceedings of the 2013 International Conference on Technological Advances in Electrical, Electronics and Computer Engineering, pages 359–364, May 2013.

[95] L. Marks, Y. Zou, and A. E. Hassan. Studying the fix-time for bugs in large open-source projects. In Proceedings of the 2011 International Conference on Predictive Models in Software Engineering, page 11. ACM, 2011.

[96] C. Weiss, R. Premraj, T. Zimmermann, and A. Zeller. How long will it take to fix this bug? In Proceedings of the 2007 International Workshop on Mining Software Repositories, page 1. IEEE Computer Society, 2007

[97] H. Zeng and D. Rine. Estimation of software defects fix effort using neural networks. In Proceedings of the 2004 Annual International Conference on Computer Software and Applications, volume 2, pages 20–21. IEEE, 2004.

[98] Q. Song, M. Shepperd, M. Cartwright, and C. Mair. Software defect association mining and defect correction effort prediction. IEEE Transactions on Software Engineering, 32(2):69–82, 2006.

[99] P. Abrahamsson, I. Fronza, R. Moser, J. Vlasenko, and W. Pedrycz. Predicting development effort from user stories. In Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement, pages 400–403.
IEEE, 2011

[100] N. C. Augen. An empirical study of using planning poker for user story estimation. In Agile Conference, 2006, pages 9–pp. IEEE, 2006.

[101] P. Abrahamsson, I. Fronza, R. Moser, J. Vlasenko, and W. Pedrycz, “Predicting development effort from user stories,” in 2011 International Symposium on Empirical Software Engineering and Measurement. IEEE, 2011, pp. 400–403.

[102] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, “Estimating story points from issue reports,” in Proceedings of the the 12th International Conference on Predictive Models and Data Analytics in Software Engineering, 2016, pp. 1–10.

[103] E. Scott and D. Pfahl, “Using developers’ features to estimate story points,” in Proceedings of the 2018 International Conference on Software and System Process, 2018, pp. 106–110.

[104] R. G. Soares, “Effort estimation via text classification and autoencoders,” in 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018, pp. 01–08.

[105] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, “A deep learning model for estimating story points,” IEEE Transactions on Software Engineering, vol. 45, no. 7, pp. 637–656, 2019.

[106] M. Abadeer, and M. Sabetzadeh. “Machine Learning-based Estimation of Story Points in Agile Development: Industrial Experience and Lessons Learned.” In 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW), pp. 106-115. IEEE, 2021

[107] L. Minku, and S. Hou. “Clustering dycom: An online cross-company software effort estimation study.” In Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering, pp. 12-21. 2017. [108] V. K. Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi. “Increasing the accuracy of software development effort estimation using projects clustering.” IET software 6, no. 6 (2012): 461-473

[109] T. Menzies, Y. Yang, G. Mathew, B. Boehm, and J. Hihn. “Negative results for software effort estimation.” Empirical Software Engineering 22, no. 5 (2017): 2658-2683.

[110] N. Bettenburg, M. Nagappan, and A.E. Hassan. “Think locally, act globally: Improving defect and effort prediction models.” In 2012 9th IEEE Working Conference on Mining Software Repositories (MSR), pp. 60-69. IEEE, 2012.

[111] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmermann. “Local vs. Global Lessons for Defect Prediction and Effort Estimation,” IEEE Trans. Software Eng., preprint, published online Dec. 2012.

[112] J. J. C. Gallego, D. Rodr´ıguez, M. A. Sicilia, M. G. Rubio, and A. G. ´ Crespo. “Software project effort estimation based on multiple parametric models generated through data clustering.” Journal of Computer Science and Technology 22, no. 3 (2007) 371-378.

[113] F. Filomena, E. Mendes, and F. Sarro. “Web effort estimation: the value of cross-company data set compared to singlecompany data set.” PROMISE 2012: 29-38.

[114] E. Mendes, M. Kalinowski, D. Martins, F. Ferrucci, and F. Sarro. “Crossvs. within-company cost estimation studies revisited: an extended systematic review.” EASE 2014: 12:1-12:10.

[115] V. Tawosi, R. Moussa, and F. Sarro. “Deep Learning for Agile Effort Estimation, Have We Solved the Problem Yet?” https://arxiv.org/abs/2201.05401, 2022.

[116] Martin Shepperd and Steve MacDonell. 2012. Evaluating prediction systems in software project estimation. Information and Software Technology 54, 8 (2012), 820–827.

[117] Muhammad Usman, Emilia Mendes, Francila Weidt, and Ricardo Britto. 2014. Effort Estimation in Agile Software

Development: A Systematic Literature Review. In Proceedings of the 10th International Conference on Predictive Models in Software Engineering (PROMISE ’14). ACM, New York, NY, USA, 82–91

[118] W AbdelMoez, Mohamed Kholief, and Fayrouz M Elsalmy. 2013. Improving bug fix-time prediction model by filtering out outliers. In Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2013 International Conference on. IEEE, 359–364.

[119] Saïd Assar, Markus Borg, and Dietmar Pfahl. 2016. Using text clustering to predict defect resolution time: a conceptual replication and an evaluation of prediction accuracy. Empirical Software Engineering 21, 4 (2016), 1437–1475.

[120] Dietmar Pfahl, Siim Karus, and Myroslava Stavnycha. 2016. Improving Expert Prediction of Issue Resolution Time. In Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE ’16). ACM, New York, NY, USA, Article 42, 6 pages

[121] Hongyu Zhang, Liang Gong, and Steve Versteeg. 2013. Predicting bug-fixing time: an empirical study of commercial software projects. In Proceedings of the 2013 international conference on software engineering. IEEE Press, 1042–1051.

[122] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Thi Minh Pham, Aditya Ghose, and Tim Menzies. 2016. A deep learning model for estimating story points. IEEE Transactions on Software Engineering (2016).

[123] Qinbao Song, Martin Shepperd, Michelle Cartwright, and Carolyn Mair. 2006. Software defect association mining and defect correction effort prediction. IEEE Transactions on Software Engineering 32, 2 (2006), 69–82.

[124] Hui Zeng and David Rine. 2004. Estimation of software defects fix effort using neural networks. In Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International, Vol. 2. IEEE, 20–21.

[125] Simone Porru, Alessandro Murgia, Serge Demeyer, Michele Marchesi, and Roberto Tonelli. 2016. Estimating Story Points from Issue Reports. In Proceedings of the the 12th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE 2016). ACM, New York, NY, USA, 2:1–2:10.

[126] Pekka Abrahamsson, Raimund Moser, Witold Pedrycz, Alberto Sillitti, and Giancarlo Succi. 2007. Effort prediction in iterative software development processes– Incremental versus global prediction models. In Empirical Software Engineering and Measurement, 2007. ESEM 2007. IEEE, 344–353.

[127] Tron Foss, Erik Stensrud, Barbara Kitchenham, and Ingunn Myrtveit. 2003. A Simulation Study of the Model Evaluation Criterion MMRE. IEEE Trans. Softw. Eng. 29, 11 (Nov. 2003), 985–995.

[128] Dan Port and Marcel Korte. 2008. Comparative Studies of the Model Evaluation Criterions MMRE and Pred in Software Cost Estimation Research. In Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM ’08). ACM, New York, NY, USA, 51–60.

[129] Barbara A Kitchenham, Lesley M Pickard, Stephen G. MacDonell, and Martin J. Shepperd. 2001. What accuracy statistics really measure. IEEE ProceedingsSoftware 148, 3 (2001), 81–85.

[130] Hoda, R., Salleh, N., & Grundy, J. (2018). The rise and evolution of agile software development. IEEE software, 35(5),

58-63

[131] Munialo, S. W., & Muketha, G. M. (2016). A review of agile software effort estimation methods. International Journal of Computer Applications Technology and Research Volume 5–Issue 9, 612-618, 2016

[132] Kaur, A., & Kaur, K. (2019). A COSMIC function points-based test effort estimation model for mobile applications. Journal of King Saud University-Computer and Information Sciences

[133] Prakash, B., & Viswanathan, V. (2017). A, "Survey on International Journal of Computer Applications (0975 – 8887)

Volume 174 – No. 13, January 2021 14 Software Estimation Techniques in Traditional and Agile Development Models". Indonesian Journal of Electrical Engineering and Computer Science, 7(3), 867-876

[134] Ahmed, M., Malik, B. H., Tahir, R. M., Perveen, S., Alvi, R. I., Rehmat, A., ... & Asghar, M. (2018, July). Estimation of Risks in Scrum Using Agile Software Development. In International Conference on Applied Human Factors and Ergonomics (pp. 111-121).

Springer, Cham

[135] Shimoda, A., & Yaguchi, K. (2017, July). A Method of Setting the Order of User Story Development of an AgileWaterfall Hybrid Method by Focusing on Common Objects. In 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAIAAI) (pp. 301-306). IEEE

[136] Adnan, M., & Afzal, M. (2017). Ontology based multiagent effort estimation system for scrum agile method. IEEE Access, 5, 25993-26005

[137] Arifin, H. H., Daengdej, J., & Khanh, N. T. (2017, March). An Empirical Study of Effort-Size and EffortTime in Expert-Based Estimations. In 2017 8th International Workshop on Empirical Software Engineering in Practice (IWESEP) (pp.
35-40). IEEE

[138] Bik, N., Lucassen, G., & Brinkkemper, S. (2017, September). A reference method for user story requirements in agile systems development. In 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW) (pp. 292-298). IEEE.

[139] Hasan & Khan. (2019). Software Development Methods – Properties and Advances. International Journal of Computer

Applications Volume 178 – No. 53

[140] Khatri, S. K., Malhotra, S., & Johri, P. (2016, September). Use case point estimation technique in software development. In 2016 5th international conference on reliability, infocom technologies and optimization (trends and future directions) (ICRITO) (pp. 123-128). IEEE.

[141] Yadav, A., & Sharma, A. (2018, May). Function Point Based Estimation of Effort and Cost in Agile Software Development. In Proceedings of 3rd International Conference on Internet of Things and Connected Technologies (ICIoTCT) (pp.
26-27).

[142] The Standish Group 2018 Chaos Report. URL: https://vitalitychicago.com/blog/agile-projects-are-moresuccessfultraditional-projects/ Retrieved on April 04, 2020

[143] Shams, A., Bohm, S., Winzer, P., & Dorner, R. (2019, July). App Cost Estimation: Evaluating Agile Environments. In 2019 IEEE 21st Conference on Business Informatics (CBI) (Vol. 1, pp. 383-390). IEEE.

[144] Gandomani, T. J., Faraji, H., & Radnejad, M. Planning Poker in cost estimation in Agile methods: Averaging Vs.

Consensus. In 2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI) (pp. 066- 071). IEEE

[145] Alstoryb, A., & Gravell, A. (2019). An empirical investigation of effort estimation in mobile apps using agile development process. Journal of Software, 14(8), 356-369

[146] III, W.G. (1987).: Generations: A Chinese Family. Richard Gordon, Carma Hinton, Kathy Kline. American Anthropologist, 89(1), pp.255–256. doi:10.1525/aa.1987.89.1.02a01150.

[147] Choetkiertikul, M., Dam, H.K., Tran, T., Ghose, A. and Grundy, J. (2018). Predicting Delivery Capability in Iterative Software Development. IEEE Transactions on Software Engineering, 44(6), pp.551–573. doi:10.1109/tse.2017.2693989 [148] Fu, M. and Tantithamthavorn, C. (2022). GPT2SP: A Transformer-Based Agile Story Point Estimation Approach.

IEEE Transactions on Software Engineering, pp.1–1. doi:10.1109/tse.2022.3158252.

[149] Niranjan, R., Kanmani, D. and Kumar, S. (2022). Design and fabrication of multi cutting hack saw machine. International journal of health sciences, pp.10294–10297. Doi: 10.53730/ijhs.v6ns3.9423

[150] Zhou, H., Chen, J., He, L., Liu, S., & Liu, C. (2020). A deep learning-based software effort estimation model. IEEE Access, 8, 141471-141483. https://doi.org/10.1109/ACCESS.2020.3014849

[151] Aljahdali, S., Alqahtani, F., & Ahmad, A. (2020). A novel feature selection approach for software effort estimation using machine learning. IEEE Access, 8, 103207-103217. https://doi.org/10.1109/ACCESS.2020.2994325

[152] Zhang, W., & Liu, X. (2020). Deep learning-based software cost estimation with dynamic feature selection. IEEE Transactions on Industrial Informatics, 16(4), 2466-2474. https://doi.org/10.1109/TII.2019.2949012

[153] Gu, M., Wang, X., & Liu, X. (2020). A deep learning approach for software cost estimation based on feature selection

and ensemble. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50(6), 2228-2239.

https://doi.org/10.1109/TSMC.2018.2884821

[154] Li, X., Li, Y., & Cao, L. (2021). Software effort estimation using deep learning with transfer learning. IEEE Transactions on Software Engineering, 47(2), 317-332. https://doi.org/10.1109/TSE.2019.2936913

[155] T., Liu, Q., & Chen, Y. (2021). A deep ensemble learning model for software effort estimation. IEEE Transactions on Software Engineering, 47(6), 1306-1322. https://doi.org/10.1109/TSE.2019.2949555

[156] Shang, Z., Li, X., & Li, Y. (2022). Deep transfer learning for software effort estimation with imbalanced data. IEEE Transactions on Software Engineering, 48(2), 141-155. https://doi.org/10.1109/TSE.2019.2936004

[157] Dong, Y., Li, X., & Li, Y. (2022). Deep learning for software development effort estimation: A comparative study.

IEEE Transactions on Software Engineering, 48(3), 311-330. https://doi.org/10.1109/TSE.2019.2945224 [158] Dhar, V., Jarke, M. and Laartz, J. (2014). Big Data. WIRTSCHAFTSINFORMATIK, 56(5), pp.277–279. doi:10.1007/s11576-014-0428-0.

[159] Flyvbjerg, B. and Budzier, A. (2011). Why Your IT Project May Be Riskier than You Think. SSRN Electronic

Journal. doi:10.2139/ssrn.2229735.

[160] Jørgensen, M. (2016). Unit effects in software project effort estimation: Workhours gives lower effort estimates than workdays. Journal of Systems and Software, 117, pp.274–281. Doi: 10.1016/j.jss.2016.03.048.

[161] Wieczorek, Ł. and Ignaciuk, P. (2018). Continuous Genetic Algorithms as Intelligent Assistance for Resource

Distribution in Logistic Systems. Data, 3(4), p.68. doi:10.3390/data3040068.

[162] Misirli, A.T., Caglayan, B., Bener, A. and Turhan, B. (2013). A Retrospective Study of Software Analytics Projects:

In-Depth Interviews with Practitioners. IEEE Software, 30(5), pp.54–61. doi:10.1109/ms.2013.93.

[163] Jorgensen, M. (2014). What We Do and Don’t Know about Software Development Effort Estimation. IEEE

Software, 31(2), pp.37–40. doi:10.1109/ms.2014.49.

[164] McConnell, S. (1998). The art, science, and engineering of software development. IEEE Software, 15(1), pp.120, 118–119. doi:10.1109/52.646892.

[165] Jorgensen, M. and Shepperd, M. (2007). A Systematic Review of Software Development Cost Estimation Studies.

IEEE Transactions on Software Engineering, 33(1), pp.33–53. doi:10.1109/tse.2007.256943.

[166] Boehm, B., Egyed, A., Port, D., Shah, A., Kwan, J. and Madachy, R. (1998). Annals of Software Engineering, 6(1/4), pp.295–321. doi:10.1023/a:1018988827405.

[167] Sentas, P., Angelis, L. and Stamelos, I. (2007). A statistical framework for analyzing the duration of software projects. Empirical Software Engineering, 13(2), pp.147–184. doi:10.1007/s10664-007-9051-7.

[168] Sentas, P., Angelis, L., Stamelos, I. and Bleris, G. (2005). Software productivity and effort prediction with ordinal regression. Information and Software Technology, 47(1), pp.17– 29. Doi: 10.1016/j.infsof.2004.05.001.

[169] Cervone, H.F. (2011). Understanding agile project management methods using Scrum. OCLC Systems & Services:

International digital library perspectives, 27(1), pp.18– 22. doi:10.1108/10650751111106528

[170] Rehman, A., Malik, A. W., Hussain, W., & Lee, Y. K. (2021). A systematic literature review of software effort estimation using machine learning. ACM Computing Surveys, 54(4), 1-43. doi: 10.1145/3450997