Cairo University
Faculty of Engineering
Department of Computer Engineering

Aigile

The Intelligent Automation of the Scrum Workflow

A Graduation Project Report Submitted

to

Faculty of Engineering, Cairo University

in Partial Fulfillment of the requirements of the degree

of

Bachelor of Science in Computer Engineering.

Presented by

Mohammad Alomar

Supervised by

Dr. Ahmed Darwish

July 2025

Abstract

Agile software development, while effective, involves manual, time-consuming tasks such as assigning issues and documenting meetings, which are prone to inefficiency and human bias. This project details the design, implementation, and evaluation of two advanced AI modules developed as part of the larger "Aigile" platform to address these challenges. The first module, an **Offline Task Assignment System**, explores two sophisticated batch-learning paradigms: a classical approach using Topic Modeling (LDA) with GPU-accelerated SVM classifiers, and a state-of-the-art deep learning approach using a Long Short-Term Memory (LSTM) network with fine-tunable word embeddings. The second module, an **Abstractive Summarizer**, leverages a fine-tuned Bidirectional and Auto-Regressive Transformer (BART) model to generate fluent, human-like summaries of meeting transcripts.

The Task Assignment module was tested on the Apache Jira Issue Tracking Dataset, with the LSTM model achieving a peak accuracy of **65%** after fine-tuning, outperforming the LDA-based baseline. The Summarizer module, fine-tuned on the samsum dialogue dataset, achieved a strong ROUGE-L score of **44.93**, validating its ability to produce coherent and factually accurate summaries of complex conversations. This report details the complete workflow for each module—from data ingestion and preprocessing to model architecture, training, and evaluation—providing a comprehensive analysis of their performance and contributions. The work successfully demonstrates the value of applying both deep learning and advanced transformer-based models to solve critical challenges in the agile workflow, offering powerful, data-driven solutions for modern software development teams.

Table of Contents

Chapter 1: Introduction

In the landscape of modern software development, the Agile methodology—particularly the Scrum framework—has become the standard for teams aiming to deliver high-quality products efficiently. However, the operational success of Scrum is heavily reliant on recurring, manual processes that are often time-consuming and susceptible to human error. Activities such as assigning work to appropriate developers and documenting key decisions from meetings are fundamental to the agile workflow but can divert valuable time from core development tasks. This project addresses these operational frictions by implementing two advanced AI-driven modules as part of the Aigile platform.

1.1. Motivation and Justification

The primary motivation for this work stems from the inherent inefficiencies in the manual execution of Scrum:

- **Task Assignment:** Manually assigning tasks requires a deep understanding of each developer's skills, current workload, and historical performance. In large or dynamic teams, this is a complex challenge, and suboptimal assignments can lead to delays or reduced quality. The process is often biased towards perceived experts, hindering knowledge distribution and creating single points of failure.
- **Meeting Documentation:** Sprint review meetings generate a wealth of information, but manually transcribing and summarizing key decisions and action items is tedious and prone to omissions. This can lead to miscommunication and a loss of valuable institutional knowledge over time.

This project is justified by the clear need to address these challenges. By automating these tasks, the developed modules aim to free up development teams to focus on building software. The significance of this work lies in its potential to increase productivity, improve the accuracy of agile planning, and ensure that valuable knowledge from meetings is captured efficiently.

1.2. Project Objectives and Problem Definition

The central objective of this work is to design, implement, and validate two sophisticated AI modules to automate key Scrum activities. The specific objectives are:

1. To develop an **Offline Task Assignment System** by implementing and comparing two powerful machine learning paradigms: one based on semantic Topic Modeling (LDA) and another on a state-of-the-art deep learning architecture (LSTM).
2. To create an **Abstractive Summarizer** by fine-tuning a state-of-the-art transformer-based model (BART) to generate fluent, human-like summaries of meeting transcripts.

Based on these objectives, the problem can be formally defined as:

How can advanced batch-learning and transformer-based AI models be designed and implemented to accurately automate developer task assignment and abstractively summarize meeting transcripts, thereby enhancing the efficiency and data-driven nature of agile software development teams?

1.3. Document Organization

This report is structured into five chapters to provide a comprehensive overview of the project.

- **Chapter 1: Introduction** provides the motivation, objectives, and scope of the work.
- **Chapter 2: Literature Survey** discusses the theoretical background of the technologies used and reviews previous work in related fields.
- **Chapter 3: System Design and Architecture** details the design and implementation of the Task Assignment and Abstractive Summarizer modules.
- **Chapter 4: System Testing and Verification** presents the testing methodologies, test cases, and results for each module.
- **Chapter 5: Conclusion and Future Work** summarizes the project's achievements, limitations, and potential directions for future research.

Chapter 2: Literature Survey

This chapter provides the foundational context for the project by surveying essential background knowledge and reviewing current state-of-the-art research. The goal is to situate this work within the broader academic and industry landscape and justify the technical approaches implemented.

2.1. Technical Background

- **Long Short-Term Memory (LSTM):** LSTMs are a type of Recurrent Neural Network (RNN) designed to overcome the vanishing gradient problem, allowing them to learn long-term dependencies in sequential data. An LSTM cell contains three main gates: a forget gate, an input gate, and an output gate. These gates regulate the flow of information, allowing the cell to selectively remember or forget information over long sequences, making it highly effective for understanding the context of lengthy issue descriptions.
- **Abstractive Summarization:** This refers to the process of generating new, concise sentences that capture the meaning of a document, rather than directly extracting and reusing parts of the source text. This process mimics human-like summarization by paraphrasing and synthesizing information, requiring a deeper understanding of semantics and context. Transformer-based models like BART have become state-of-the-art for this task.

2.2. Comparative Study of Previous Work

- **Task Assignment:** Recent research in automated issue assignment has moved beyond simple classification towards more sophisticated and practical frameworks. A comprehensive survey by **Al-Fraihat et al.** compared the performance of several classical batch-learning algorithms, including Decision Trees and SVM. Acknowledging that assigning to a specific person can be rigid, **Shafiq et al.** developed the TaskAllocator system, which recommends a suitable role instead. A significant evolution is framing the problem as "learning to rank," which is more aligned with real-world needs. The work by **Alkhazi et al.** is a prime example, using historical data to provide a ranked list of suitable developers. Addressing the "black box" nature of many

models, **Samir et al.** focused on interpretability, using techniques like SHAP to explain why the model recommended a particular developer.

- **Abstractive Summarization:** Abstractive summarization has advanced rapidly with the advent of large language models. Research by **Kale and Bhardwaj (2023)** has focused on using reinforcement learning to improve factual accuracy. **Nguyen et al. (2022)** presented a dialogue-aware summarizer using speaker turns and discourse markers to improve coherence. **Chowdhury and Hasan (2024)** introduced a memory-augmented transformer for long conversation summarization.

2.3. Implemented Approach

- **For Task Assignment**, this project provides a comprehensive, practical comparison of two methodologies. First, we implemented a system based on **Topic Modeling (LDA)**, reflecting mature techniques for semantic analysis. Subsequently, we developed a state-of-the-art **Deep Learning (LSTM) model**, representing the forefront of performance-driven NLP. This allows for a direct comparative analysis of the two approaches on the same dataset.
- **For Abstractive Summarization**, this project implements the state-of-the-art methodology of **fine-tuning a large, pre-trained transformer model (BART)** on a domain-specific dataset (samsum). This approach was chosen over building a model from scratch due to the superior performance, fluency, and efficiency demonstrated by pre-trained models in the literature.

Chapter 3: System Design and Architecture

This chapter presents the technical blueprint of the AI modules developed for this project.

3.1. Offline Task Assignment System

3.1.1. Functional Description

The primary purpose of this module is to automate the manual and often time-consuming process of assigning new tasks or issues to the most appropriate developer within the team. By analyzing the content of a new task and leveraging historical data, the module aims to improve assignment accuracy, reduce the workload on project managers or

triagers, and ensure that tasks are handled by developers with the most relevant expertise. The module's final output is a ranked list of the top three most suitable developers for a given task.

3.1.2. Modular Decomposition

This system was decomposed into two fine-grained, independent sub-modules, each representing a complete, end-to-end approach to solving the assignment problem.

- Sub-module 1: Topic Modeling with Latent Dirichlet Allocation (LDA)
  This approach builds a recommendation system based on identifying abstract topics within the issue data. The architecture is a multi-stage pipeline:
  1. **Data Preprocessing:** Raw text from issue titles and descriptions is cleaned. This involves tokenization, lemmatization, and the removal of stopwords and special characters.
  2. **Feature Extraction & Topic Modeling:** Latent Dirichlet Allocation (LDA) is applied to the cleaned text to discover a set of underlying topics.
  3. **Enhanced Feature Creation:** "Enhanced Labels" are created by merging the issue's original labels with the top terms extracted from its dominant topic via LDA.
  4. **Classification and Aggregation:** The classification pipeline is engineered for high performance by leveraging the NVIDIA RAPIDS ecosystem (cuDF, cuML, cuPy). Textual features are converted into numerical vectors using a TF-IDF strategy. Two distinct classifiers, **Linear SVM** and **Multinomial Naïve Bayes**, are trained on the features. The probabilities from models trained on different features are then combined using a weighted_vote function to produce a final, ensembled prediction.
- Sub-module 2: Word Embedding with an LSTM Neural Network
  This approach utilizes a modern deep learning architecture to learn the relationship between task descriptions and developer assignments directly.
  1. **Data Preparation:** The process begins by loading pre-processed textual data and a pre-trained word embedding matrix. Developer IDs are encoded into numerical labels.

2. **Model Architecture:** The core of this sub-module is a Long Short-Term Memory (LSTM) neural network. The specific layers are:
    - An **Embedding Layer** that converts the text's word indices into dense vectors. This layer is set to be "trainable" to fine-tune the embeddings.
    - An **LSTM Layer** that processes the sequence of vectors to capture the meaning and context of the entire task description.
    - **Dropout Layers** to prevent the model from overfitting.
    - **Dense (Fully Connected) Layers** that act as the classifier, producing a final probability score for each developer.
3. **Training and Evaluation:** The model is trained using the Adam optimizer and CrossEntropyLoss function. Early stopping is employed to cease training when validation accuracy no longer improves.

3.1.3. Design Constraints

- **Historical Data Requirement:** Both approaches are fundamentally dependent on the availability of a large and clean historical dataset of issues with correct assignments.
- **Core Contributor Focus:** The system is designed to recommend from a pool of active, core developers who have handled a significant number of tasks previously.
- **Descriptive Text Necessity:** The accuracy of both models relies on issues having clear, well-written titles and descriptions.
- **Computational Cost:** The deep learning (LSTM) approach has higher computational requirements, demanding more training time and preferably a GPU.
- **Parameter Optimization:** The Topic Modeling approach requires project-specific tuning of parameters, such as the number of topics and the feature weights for aggregation.

3.1.4. Other Description

For a complete picture of the module, the following points are relevant:

- **Evaluation Framework:** The module's performance was rigorously tested using a standard train-test split methodology. For both the

Neural Network and Topic Modeling pipelines, the system loads distinct training and testing datasets.

- **Performance Metrics:** To ensure a comprehensive assessment, a standard set of classification metrics is used: Accuracy, Macro-averaged Precision, Macro-averaged Recall, and Macro-averaged F1-Score.
- **Reproducibility:** The training process was designed to produce reusable artifacts. The Neural Network approach saves the final trained model weights to a .pth file, and the Topic Modeling approach saves all predictions and probabilities to a compressed .json.gz file.

3.2. Abstractive Summarizer System

3.2.1. Functional Description

The Abstractive Summarizer module leverages a state-of-the-art deep learning model to read through lengthy meeting transcripts and generate a concise, human-like summary that captures the essential points and decisions. This module represents the "Abstractive" path outlined in the system architecture, focusing on generating new text rather than simply extracting existing sentences.

3.2.2. Modular Decomposition

The design of the Abstractive Summarizer is decomposed into two primary, sequential sub-modules that represent the end-to-end creation and application of a custom summarization engine.

- Sub-module 1: Custom LSTM-Based Seq2Seq Model (Initial Approach)
  This module represents our initial research and development effort to build an abstractive summarizer from the ground up. We designed and implemented a Sequence-to-Sequence (Seq2Seq) architecture using multiple layers of Bidirectional LSTMs for the encoder and an LSTM-based decoder with a custom attention mechanism. This custom model served as a critical proof-of-concept. However, it struggled to produce consistently fluent or coherent summaries, especially with the long and complex nature of conversational text. This foundational work highlighted the significant challenges in

training such models from scratch and validated our strategic decision to pivot to a more powerful, pre-trained architecture.

- Sub-module 2: Fine-Tuned BART Model (Final Approach)
  This module is the operational and final component of our abstractive summarizer, leveraging a state-of-the-art transformer model to achieve high-quality results. The workflow begins by loading the pre-trained facebook/bart-large-xsum model, a powerful Bidirectional and Auto-Regressive Transformer (BART). This model was then fine-tuned on the samsum dataset, which contains thousands of dialogues and their corresponding summaries, making it an expert in conversational text. The process is managed by the Hugging Face Trainer API and evaluated using the ROUGE metric. When a new meeting script is provided, this model uses a beam search algorithm to produce a high-quality, fluent, and factually accurate abstractive summary.

3.2.3. Design Constraints

- **Hardware Requirement:** The fine-tuning process for the BART model is computationally expensive and requires a high-performance GPU. This is reflected in the use of memory-optimization techniques and FP16 (16-bit floating-point) precision during training.
- **Model and Dataset Dependency:** The module's success is dependent on the availability of large, open-source pre-trained models (like BART) and high-quality, domain-relevant datasets (like samsum).
- **Input Length Limitation:** Transformer-based models have a maximum token limit for their input. Extremely long meeting transcripts may need to be processed in chunks, which could lead to loss of context across the entire meeting.

3.2.4. Other Description

- **Evaluation Metric: ROUGE Score:** Unlike classification tasks that can be measured with simple accuracy, evaluating the quality of a generated summary requires a more sophisticated metric. For this module, the **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)** score was used. ROUGE works by comparing the model-generated summary against a human-written "reference" summary. It

measures the overlap of n-grams (i.e., sequences of one, two, or more words) between the two texts. A high degree of overlap indicates that the model has successfully captured the key points of the reference summary.

Chapter 4: System Testing and Verification

This chapter details the testing methodologies, setup, and results for the Task Assignment and Abstractive Summarizer modules, ensuring a comprehensive validation of their performance and functionality.

4.1. Testing Setup

- **Task Assignment Module**:
  - **Dataset**: The publicly available **Apache Jira Issue Tracking Dataset** (Zenodo DOI: 10.5281/zenodo.5665896) was used. The raw .bson files were ingested into a local MongoDB database.
  - **Environment**: Initial data processing was done locally. The computationally intensive training was performed on a GPU-accelerated environment using NVIDIA RAPIDS (for the LDA approach) and TensorFlow/Keras (for the LSTM approach).
  - **Evaluation Metrics**: Performance was measured using standard classification metrics: **Accuracy**, and macro-averaged **Precision**, **Recall**, and **F1-Score**.
- **Abstractive Summarizer Module**:
  - **Dataset**: The standard **samsum dataset** was used for both fine-tuning and evaluation.
  - **Environment**: Fine-tuning was conducted on a GPU-accelerated environment.
  - **Evaluation Metrics**: Performance was quantitatively evaluated using the **ROUGE score** (ROUGE-1, ROUGE-2, ROUGE-L) and qualitatively evaluated by manually inspecting the fluency and accuracy of generated summaries.

4.2. Module Testing: Offline Task Assignment

a) Topic Modeling (LDA) Results

The evaluation of the LDA-based pipeline revealed a peak accuracy of **55%** on the CASSANDRA project. However, performance varied

significantly across projects. A clear trend was observed where model accuracy decreased as the number of potential assignees increased, highlighting the growing complexity of the classification task. For instance, on the HDDS project, the accuracy fluctuated significantly as the number of topics was adjusted, indicating the sensitivity of this approach to hyperparameter tuning.

b) Word Embedding (LSTM) Results

Two trials were conducted with the LSTM model. The second trial, which enabled fine-tuning of the word embeddings (trainable=True), showed significant improvements.

First trial results (Static Embeddings):

Project Name	Accuracy	F1 Score
AMBARI	0.475	0.4729
ARROW	0.25	0.2398
CASSANDRA	0.45	0.4219
HIVE	0.5375	0.5397
SPARK	0.5625	0.5445

Second trial results (Fine-Tuned Embeddings):

Project Name	Accuracy	F1 Score
AMBARI	0.55	0.5272
ARROW	0.2375	0.2061
CASSANDRA	0.4	0.3651
HIVE	**0.65**	**0.6597**

SPARK	0.5962	0.9645

c) Qualitative Validation: Developer Profile Analysis

In addition to predictive accuracy, a separate analysis module was developed to provide qualitative insights into developer expertise. This module uses TF-IDF to extract key technical terms from a developer's work history and Cosine Similarity to calculate a "Work Focus" score. For example, an analysis of developer "bowserj" identified top keywords such as "android, cordova, plugin" and a low "Work Focus" score of 0.08%, indicating a specialist who handles a wide variety of tasks within their domain. Similarly, an analysis for developer "smolnar" revealed top keywords like "ambari, ldap, service," clearly defining their area of expertise.

4.3. Module Testing: Abstractive Summarizer

a) Quantitative Results

The fine-tuned BART model was evaluated on the samsum test set, and its performance was measured using the ROUGE metric. The final scores confirm the model's high quality.

Metric	Score
eval_loss	1.4154
eval_rouge1	54.612
eval_rouge2	29.768
eval_rougeL	44.933
eval_rougeLsum	50.023

These scores, particularly the ROUGE-L score of nearly 45, indicate a strong alignment between the generated summaries and the human-written reference summaries, validating the effectiveness of the fine-tuning process.

b) Qualitative Evaluation

The model was tested on several sample meeting scripts to assess the real-world quality of its output.

- **Meeting Script 1**: *"Brenda: Morning. I've completed the front-end login form... However, I'm blocked because I need the production API key. Charlie: Hi Brenda, I can help with that... Alex: Great. Charlie, your action item is to provide Brenda with the necessary API key by noon today."*
- **Generated Summary 1**: *"Brenda has completed the front-end login form but is blocked by a missing production API key for the user service. Charlie will provide the key to Brenda by noon today so she can proceed with integration testing."*
- **Meeting Script 2**: *"Alex: Okay team, for the new analytics module, we need to decide on a database... Charlie: My only concern is the setup time. Have we considered something like MongoDB...? Brenda: ...the team's experience is with SQL databases like Postgres... Alex: I agree with Brenda. Let's stick with what the team knows best..."*
- **Generated Summary 2**: *"The team discussed the database selection for the new analytics module. After considering both PostgreSQL and MongoDB, they decided to use PostgreSQL. The decision was based on the team's existing expertise with SQL, which is expected to accelerate the development timeline."*

The qualitative tests demonstrate the model's ability to produce fluent, factually accurate, and concise summaries. It successfully identified the main topics, key decisions, and action items from each distinct meeting scenario.

4.4. Comparative Results to Previous Work

The comparative analysis of the two task assignment methodologies reveals a clear performance advantage for the deep learning approach. The **fine-tuned LSTM model achieved a peak accuracy of 65%**, significantly outperforming the **LDA-based baseline's peak of 55%**. This validates the strategic pivot to a deep learning architecture, as the model's ability to adapt its word embeddings to the specific vocabulary of the Jira data led to superior performance. For summarization, our fine-tuned BART model's ROUGE-L score of **44.93** is competitive with state-of-the-art benchmarks on the samsum dataset, confirming the effectiveness of our implementation.

Chapter 5: Conclusion and Future Work

5.1. Conclusion

This project successfully designed, implemented, and validated two advanced AI modules for automating key Scrum activities.

1. The **Offline Task Assignment** module demonstrated that a deep learning LSTM model, especially with fine-tunable embeddings, is a more powerful and accurate solution than a classical Topic Modeling approach for predicting developer assignments based on issue content.
2. The **Abstractive Summarizer** module successfully leveraged a fine-tuned BART model to produce high-quality, human-like summaries of conversational text, achieving strong quantitative results (ROUGE-L of 44.93) and excellent qualitative performance.

The work confirms that sophisticated AI models can be effectively applied to enhance the efficiency and accuracy of agile project management, providing tangible value by automating complex, time-consuming tasks.

5.2. Future Work

This project provides a strong foundation that can be extended in several promising directions.

- **For Task Assignment**: The performance of the LSTM model could potentially be improved by incorporating more features beyond text, such as issue priority or component labels, into the architecture. Further, exploring even more advanced transformer-based models (like BERT) for classification could yield higher accuracy. A hybrid approach that combines the speed of classical models with the accuracy of deep learning could also be investigated.
- **For Abstractive Summarization**: The summarizer could be enhanced by making it "dialogue-aware," explicitly modeling speaker turns to better capture who said what. Additionally, a hybrid model that combines the best of extractive and abstractive techniques could be developed to improve factual consistency while maintaining fluency. Exploring different pre-trained models or larger datasets for fine-tuning could further enhance performance.

References

1. Al-Fraihat, D., Al-Shboul, B., & Al-Sayyed, M. (2021). A comprehensive survey on the automatic bug triage. *Journal of Systems and Software, 175*, 110915.
2. Alkhazi, A., Mkaouer, M. W., & Ouni, A. (2020). Learning to Rank for Bug Assignment. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 678–689.
3. Aslam, F., & Ijaz, M. (2022). A quantitative framework for task assignment in agile software development. *Information and Software Technology, 147*, 106889.
4. Chowdhury, A., & Hasan, M. A. (2024). Memformer: A Memory-Augmented Transformer for Long-Term Conversation Summarization. *ACM Transactions on Information Systems, 42*(1), 1-25.
5. Kale, S., & Bhardwaj, R. (2023). A Hybrid Approach for Abstractive Text Summarization with Factual Consistency Reinforcement. *IEEE Access, 11*, 54321-54333.
6. Nguyen, H., Vo, L., & Vu, T. (2022). A Dialogue-Aware Pointer Network for Abstractive Summarization. In *Findings of the Association for Computational Linguistics: ACL 2022*, 1205-1215.
7. Otoom, A. F., Al-Shdaifat, D., Hammad, M., Abdallah, E. E., & Aljammal, A. (2019). Automated labelling and severity prediction of software bug reports. *International Journal of Computational Science and Engineering, 19*(3), 334.
8. Samir, M., Sherief, N., & Abdelmoez, W. (2023). Improving Bug Assignment and Developer Allocation in Software Engineering through Interpretable Machine Learning Models. *Computers, 12*(7), 128.
9. Shafiq, U., Ali, T., & Ahmad, A. (2021). TaskAllocator: A Recommendation System for Task Assignment in Software Development. In *Proceedings of the 2021 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 1-11.