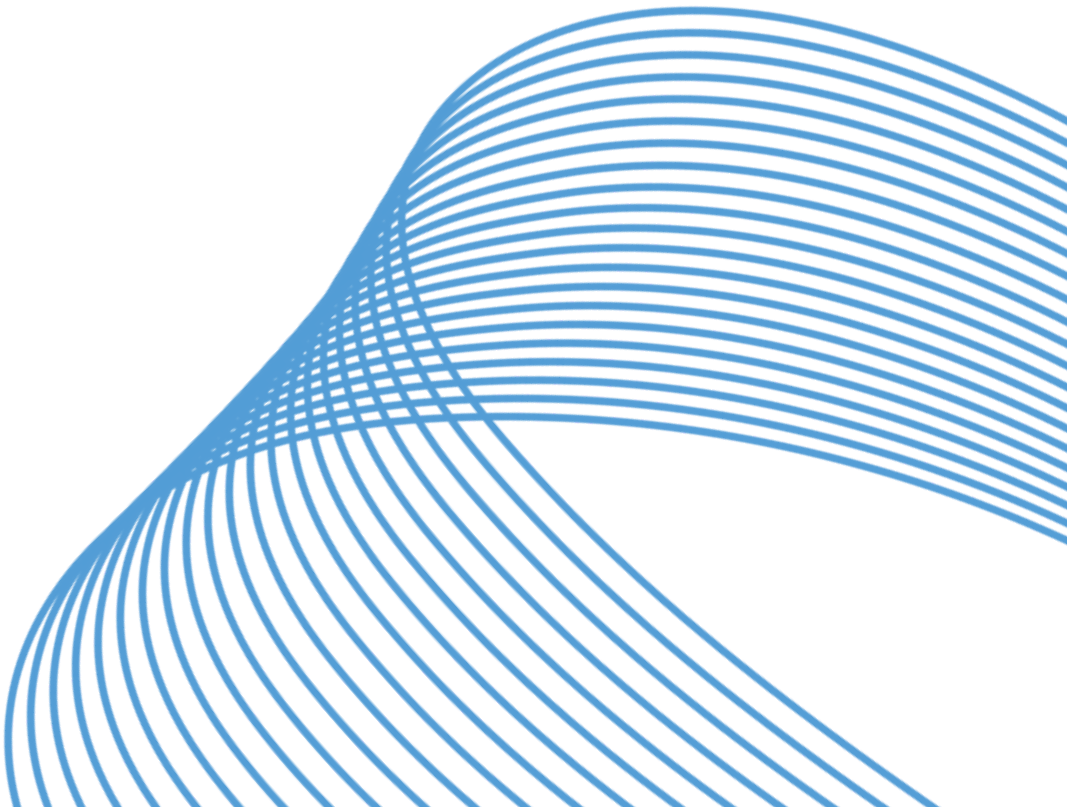
TU Delft

ENHANCING ISSUE TRACKING EFFICIENCY WITH AI-DRIVEN NATURAL LANGUAGE PROCESSING: IMPROVING CLASSIFICATION, ASSOCIATION AND RESOLUTION

MASTER THESIS BY

VENELINA POCHEVA



Enhancing Issue Tracking Efficiency with AI-Driven Natural Language Processing: Improving Classification, Association and Resolution

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Venelina Pocheva
born in Ruse, Bulgaria

Software Engineering Research Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

NXP Semiconductors B.V.
High Tech Campus 60, 5656 AG
Eindhoven, the Netherlands
https://www.nxp.com/

Enhancing Issue Tracking Efficiency with AI-Driven Natural Language Processing: Improving Classification, Association and Resolution

Author: Venelina Pocheva
Student id: 5093570

Abstract

In large-scale engineering environments, efficient issue tracking is essential for timely problem resolution and knowledge reuse. However, the manual classification and association of issue reports present scalability challenges, further complicated by inconsistent annotations and the absence of semantic linking mechanisms. This project investigates the application of Natural Language Processing and Artificial Intelligence to automate multi-label classification and discover meaningful semantic associations between technical issues. Over 70 model configurations were evaluated on a real-world industrial dataset, comparing classical models with transformer-based and deep learning approaches. DistilBERT achieved the highest Recall@5 (0.93), indicating strong performance in identifying relevant categories. Classical methods, such as TF-IDF combined with Logistic Regression, also performed well, offering a computationally efficient and interpretable option. For association discovery, approaches including lexical retrieval, embedding-based similarity, clustering-based filtering, and topic modelling were assessed using both quantitative metrics and expert review. Lexical (BM25) and embedding-based (SBERT + Cosine Similarity) methods offer complementary strengths, retrieving overlapping yet distinct sets of associations. Associations identified by both models were rated as useful in over 70% of cases by domain experts, suggesting that agreement between methods may serve as an indicator of relevance. While Copilot provided consistent relevance assessments, its ratings were often higher than those provided by human evaluators and did not always reflect their detailed assessments. These findings highlight the potential of combining lexical and semantic methods with human-in-the-loop validation to support scalable and accurate industrial applicability.

Thesis Committee:

Chair:	Dr. Neil Yorke-Smith
University supervisor:	Dr. Maliheh Izadi
Company supervisor:	René van den Berg
Committee Member:	Dr. Andreea Costea

Preface

This thesis marks the final step in my Master's in Computer Science, and I would like to express my sincere appreciation to everyone who supported me during this journey.

I am grateful to the academic staff and supervisors at Delft University of Technology for their guidance, feedback, and encouragement throughout the project. I also extend my thanks to the colleagues and mentors at NXP Semiconductors, whose input and support were instrumental in shaping the direction and relevance of this work.

I am especially thankful for the opportunities to learn from experts across both academic and industrial settings. Their support and feedback were invaluable throughout the process.

I would like to thank my friends for their encouragement along the way. I am also thankful to my boyfriend for his understanding, support and for being both a welcome distraction and a source of motivation throughout this journey. Most of all, I am deeply grateful to my family for their unconditional love, unwavering belief in me, and constant support through every step of this path.

Venelina Pocheva
Delft, the Netherlands
July 16, 2025

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long Short-Term Memory
BM25	Best Matching 25
BoW	Bag-of-Words
BR	Binary Relevance
CC	Classifier Chain
CNN	Convolutional Neural Network
CRISP-DM	Cross-Industry Standard Process for Data Mining
CV	Count Vectorizer
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DistilBERT	Distilled BERT
EDA	Exploratory Data Analysis
F1	Harmonic mean of precision and recall
GNB	Gaussian Naive Bayes
IoT	Internet of Things
kNN	k-Nearest Neighbour
LDA	Latent Dirichlet Allocation

LLM	Large Language Model
LinearSVC	Linear Support Vector Classification
LP	Label Powerset
LR	Logistic Regression
MAP	Mean Average Precision
ML	Machine Learning
ML-kNN	Multi-Label k-Nearest Neighbours
MLC	Multi-label Classification
MLP	Multilayer Perceptron
MNB	Multinomial Naive Bayes
MMP	Multiclass Multilabel Perceptron
MPP	Mean Proportion of Properly Predicted Labels
NLP	Natural Language Processing
OvR	One-vs-Rest
RC	Ridge Classifier
RF	Random Forest
RNN	Recurrent Neural Network
RQ	Research Question
RankSVM	Ranking Support Vector Machine
RoBERTa	Robustly Optimised BERT Approach
SBERT	Sentence Bidirectional Encoder Representations from Transformers
SMOTE	Synthetic Minority Over-sampling Technique
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
TN/FN	True Negative/False Negative
TP/FP	True Positive/False Positive
XGBoost	Extreme Gradient Boosting

Chapter 1

Introduction

1.1 Overview

In an era where data is frequently described as *"the new oil"*, the principal challenge lies not in its acquisition but rather in its interpretation, comprehension, and effective utilisation [56]. Engineering environments, particularly engineering issue tracking systems, exemplify this challenge significantly. The rapid advancement of technology has led to an overwhelming volume of unstructured data, generated during engineering workflows, through issue logs, defect reports, and project documentation. These textual data records are essential for capturing technical problems, their analyses, and the corresponding solutions [66]. However, their unstructured nature makes it difficult to systematically extract actionable insights, leading to a growing gap between data availability and its effective use [36].

At NXP Semiconductors, a global leader in embedded systems and secure connectivity that operates across industries including automotive, industrial automation, and the Internet of Things (IoT) [41], this challenge is particularly pronounced. As software and hardware development workflows grow in scale and complexity, the company faces challenges in efficiently managing its large-scale issue tracking systems. Traditional management approaches rely heavily on manual operations. Engineers are required to classify and associate issues manually, a process that becomes increasingly unsustainable as the volume and complexity of data expand [3]. Moreover, manual classification introduces inconsistency, is prone to human error, and often fails to capture complex relationships between issues, particularly in interdisciplinary projects [1]. Furthermore, engineers often rely on past experience to detect recurring patterns or link related issues, leading to duplicated efforts and missed opportunities for early problem detection. These challenges underscore the necessity for scalable, automated solutions capable of augmenting or replacing manual interventions.

To address these inefficiencies, Machine Learning (ML) techniques – particularly those in the area of Natural Language Processing (NLP) – have emerged as promising solutions within the broader field of Artificial Intelligence (AI). NLP techniques are designed to extract structured information from unstructured text and have been successfully applied to tasks such as classification, association discovery, and information retrieval [14, 16, 45]. In

engineering contexts, these applications can reduce manual effort, minimise errors, enhance productivity and support improved knowledge reuse [59].
In this context, the present study focuses on exploring how NLP and ML can be leveraged to address these problems in the current issue management practices [1]. By implementing and evaluating AI-driven techniques tailored to engineering workflows, the research seeks to enhance the scalability, efficiency, and usability of issue tracking system. Ultimately, this work aims to contribute to more effective knowledge management, faster problem resolution, and improved engineering outcomes.

1.2 Problem Statement

Effective issue tracking is a critical component in managing large-scale engineering projects, particularly in high-tech industries such as semiconductors [46]. The current issue management system faces limitations that hinder both efficiency and knowledge reuse across engineering teams and different projects. While the existing process supports structured documentation and traceability, it lacks automation in key areas, which affects the quality and speed of issue handling.
One major inefficiency concerns the **classification** of newly logged issues. In the current workflow, each issue is manually assigned a single category corresponding to the team responsible for addressing it. As the issue resolution progresses, the category is overwritten to reflect the next team involved; however, the system does not maintain a complete record of the resolution path. Consequently, historical traceability is lost, and it becomes difficult to understand how issues move through the organisation. Additionally, there is no standardised taxonomy across teams – each team defines categories independently, resulting in inconsistencies in spelling, abbreviations, and formatting. These inconsistencies, combined with the one-category limitation, prevent multiple relevant teams from being notified early in the process about the issue and hinder opportunities for possible parallel problem-solving. As a result, the classification data becomes less useful for coordination, trend analysis, and automated routing.
Another key inefficiency is related to **association discovery**. Relationships between issues, such as recurring faults and related problems, are not systematically identified. Manual linking is possible, but often reflects procedural connections, such as demonstration-related tasks or follow-up actions, rather than deeper semantic relationships that could support problem-solving and continuous improvement. As a result, opportunities for reusing historical information and knowledge, clustering of similar issues, or early identification of patterns are frequently missed.
These limitations pose several risks: delays in resolution, unnecessary duplication of effort, and reduced visibility into systemic engineering problems. To address these challenges, this study investigates techniques to automate classification and discover meaningful asso-

[1] This research focuses only on textual data from the issue tracking system *Collabnet* (titles, descriptions, and manually assigned metadata); it does not cover the analysis of non-textual data sources such as sensor logs or code repositories. The scope is limited to the investigation and evaluation of ML/AI techniques; full deployment and integration into production systems are considered out of scope for this project.

ciations within industrial issue reports, ultimately aiming to improve the speed, accuracy, and scalability of the issue tracking processes.

1.3 Objective

The primary objective of this research is to design and implement solutions driven by AI to enhance the issue tracking system. The study specifically focuses on automating the classification of issues through multi-label learning and introducing a mechanism for discovering semantic associations between related reports. By achieving these objectives, the research aims to increase the efficiency, accuracy, and scalability of issue management within the engineering workflows.

1.3.1 Research Questions

The research questions investigated in this project are:

RQ: How can AI-driven techniques be leveraged to automate and enhance the classification and association of industrial issue reports?

RQ1: How can multi-label classification techniques be effectively applied to categorise industrial issue reports across a large and diverse set of categories?

RQ2: How can semantic associations between issue reports be identified using NLP techniques, and how can the most relevant related issues be retrieved for a given report?

1.4 Research Process

To answer these research questions, the following tasks have been completed:

1. First, a comprehensive literature review was conducted to examine existing NLP techniques and broader ML/AI methods, with a particular focus on their applicability to engineering issue tracking systems.
2. Following this, data was collected and preprocessed from the issue tracking database, in compliance with data privacy and confidentiality requirements.
3. Exploratory data analysis (EDA) was performed to identify patterns, trends, and correlations within the dataset.
4. Several models for issue classification were developed and evaluated using appropriate performance metrics to assess their effectiveness.
5. Techniques for association discovery were implemented to identify semantic relationships between related issues.
6. Finally, the performance of the implemented models and techniques was compared, and insights were used to support recommendations for potential integration into the issue management system.

1.5 Contributions

The following contributions have been made as part of this thesis:

1. **Design and evaluation of a multi-label classification pipeline** for engineering issue reports, comparing classical models with transformer-based models, and highlighting trade-offs in performance, complexity, and deployment feasibility.
2. **Evaluation of association discovery methods**, including semantic similarity, topic modelling, lexical retrieval, and scoring-based approaches, applied to a large-scale industrial dataset.
3. **Design of a human evaluation framework** for association relevance, involving the generation of top-5 recommendations for each method and expert-based scoring, allowing comparison despite the absence of a complete ground truth.
4. **Empirical analysis of inter-rater agreement and model performance**, based on expert-based evaluations and statistical testing, to assess consistency across raters, significance of performance differences, and robustness of association ranking results.
5. **Direct comparison between Large Language Model (LLM) and expert evaluations**, quantifying agreement, highlighting divergence patterns, and assessing the reliability of LLMs as evaluators in industrial NLP applications.
6. **Practical recommendations for the deployment of NLP solutions in industrial issue tracking systems**, based on performance trends, human evaluations, and considerations such as scalability, interpretability, and integration constraints.
7. **Applied research contribution** through a systematic evaluation of existing NLP and ML techniques on large-scale industrial issue tracking data. While this thesis does not propose novel algorithms, it examines the practical applicability and limitations of these techniques.

1.6 Thesis Structure

This thesis is structured into six chapters, each addressing a key component of the research. Chapter 2 presents the background and related work relevant to this thesis, organised around its two primary tasks: multi-label classification and association discovery. Chapter 3 outlines the research methodology, which was based on the CRISP-DM (Cross Industry Standard Process for Data Mining) framework to guide the systematic exploration and evaluation of the proposed techniques. Chapters 4 and 5 detail the experimental work conducted, focusing, respectively, on automating classification and association discovery. Each of these chapters follows a consistent structure, including a problem definition, methodology, results, and discussion. Finally, Chapter 6 concludes the thesis by summarising the main findings and contributions, and by providing recommendations for future research directions.

Chapter 2

Related Work and Background

This chapter reviews the methodologies relevant to this study, focusing on two primary tasks: multi-label issue classification and association discovery. For each task, foundational principles are introduced, followed by an overview of recent approaches commonly used in academic and applied settings. The chapter concludes with a summary of key insights and identification of research gaps that motivate this thesis.

2.1 Classification

Text classification is a fundamental task in NLP, typically involving the assignment of one or more predefined categories to a given text. While traditional classification assumes that each document belongs to a single category, many real-world tasks involve multiple relevant labels. This setting is modelled as a multi-label classification (MLC) problem, where each instance can be associated with a subset of categories from a larger label space [21]. This becomes especially important in industrial domains, such as engineering issue tracking, where issues often belong to multiple categories simultaneously. In the context of this study, MLC provides a more realistic modelling approach for classifying engineering issues that span multiple teams, subsystems, or problem types.

Compared to single-label classification, MLC introduces several unique challenges that significantly impact model design and performance. First, **label imbalance** is common – some categories are heavily underrepresented in the training data. This can lead to biased models that overfit to frequent labels and neglect rare but important ones. Second, **label co-occurrence** sparsity arises because not all combinations of labels appear in the training set, especially in high-dimensional label spaces. This limits a model's ability to generalise to unseen label combinations. Third, models must account for **label dependencies**: some labels tend to occur together, and treating labels as independent can lead to inconsistent or unrealistic predictions. Finally, **evaluation** in MLC is more complex than in the single-label setting, as performance must reflect both exact and partial correctness of predictions. Metrics like Hamming loss, F1-score, Precision@k, Recall@k and ranking-based measures are commonly used to capture different aspects of performance [15, 57].

Together, these challenges motivate the use of dedicated MLC approaches that can model

label dependencies, handle class imbalance, and adapt to partial or sparse supervision. Broadly, the literature organises such methods into two main categories: **problem transformation** techniques, which reformulate MLC as multiple simpler learning tasks, and **algorithm adaptation** methods, which extend base learning algorithms to handle multi-label data natively. The following sections review both categories in more detail.

2.1.1 Problem Transformation

Problem transformation methods tackle multi-label classification by breaking it down into simpler tasks. Rather than predicting multiple labels all at once, they transform the problem into several single-label or binary classification tasks, each handled by a standard machine learning model. These methods are attractive because they allow the use of standard classifiers with minimal modification. However, they may struggle to capture complex interdependencies between labels [5, 21].

In practice, problem transformation methods are often implemented using classical machine learning models such as Logistic Regression, SVM, Ridge Classifier, Random Forest, and XGBoost. These models are favoured for their computational efficiency and compatibility with sparse feature vectors like TF-IDF or Bag-of-Words [21].

The most basic problem transformation strategy is **Binary Relevance (BR)**, which treats each label as an independent binary classification problem. A separate classifier is trained for each label, predicting its presence or absence for a given instance. Although BR is conceptually simple and computationally efficient, it does not account for label correlations, which may result in incoherent label sets for interdependent categories [21, 57].

A related method is **One-vs-Rest (OvR)**, often used when base classifiers (e.g., SVM) are inherently multi-class. Like BR, OvR does not model label dependencies but can be more effective when label-specific decision boundaries are well separated [57].

To address the lack of label dependency modelling, **Classifier Chains (CC)** have been proposed. In CC, classifiers are arranged in a sequence, and each classifier receives the predicted labels of previous classifiers as additional input features. This allows the model to learn conditional dependencies between labels. However, the performance of CC is sensitive to the chain order and is susceptible to error propagation if early classifiers make incorrect predictions [21].

Another widely used approach is **Label Powerset (LP)**, which transforms the MLC problem into a single multi-class classification task by treating each unique combination of labels as a separate class. LP captures label dependencies effectively but becomes computationally infeasible when the number of distinct label combinations grows, which is common in high-dimensional label spaces [21].

Arslan and Cruz [4] demonstrate that even simple problem transformation methods like Binary Relevance can be effective in real-world scenarios, outperforming more complex approaches such as Classifier Chains and Label Powerset on imbalanced business datasets with moderately large label spaces. Their findings suggest that when label dependencies are limited, the simplicity and scalability of BR can be advantageous. Bogatinovski et al. [5] report that CC and LP outperform BR in datasets with strong label interdependencies, while Chalkidis et al. [8] demonstrate the effectiveness of LP-based deep models in large-scale

legal text classification. Han et al. [21] provide theoretical support for these results, emphasising that modelling label correlations improves predictive performance. Moreover, Tarekegn et al. [57] highlight that transformation methods remain effective even under class imbalance, particularly when combined with resampling strategies. Veeranki et al. [59] further validate the practical utility of CC in clinical multi-label classification tasks. Collectively, these studies suggest that problem transformation methods remain versatile and competitive across a range of domains, with their success depending on the structure of the label space and task complexity.

2.1.2 Algorithm Adaptation

Instead of decomposing the multi-label classification problem into multiple single-label tasks, algorithm adaptation methods modify learning algorithms to natively handle multi-label outputs. These techniques are especially useful when dealing with large output spaces or when greater expressiveness is needed from the classifier.

One widely used method is **ML-kNN**, a learning algorithm that extends the traditional k-nearest neighbours by estimating label relevance based on frequency counts among neighbouring instances [67]. Han et al. [21] show that ML-kNN achieves strong performance on benchmark datasets like Yeast and Scene, often outperforming BR due to its ability to consider joint label distributions. A recent comparison by Xu et al. [65] further confirms ML-kNN's comparative performance, especially on datasets with a moderate number of labels and limited training data, where instance-based reasoning remains effective.

Another influential adaptation is **RankSVM**, which frames MLC as a ranking problem and optimises the margin between relevant and irrelevant labels. Joachims [23] originally proposed this approach for information retrieval, and subsequent adaptations have shown it to be effective for MLC tasks, especially where label importance or priority matters. Despite its computational cost, RankSVM performs well in cases where ranking between labels is critical. In a comparative evaluation by Han et al. [21], RankSVM demonstrates favourable ranking performance, although its scalability is more limited than other methods.

An important direction in algorithm adaptation is the use of margin-based structured prediction for multi-label ranking. Ghamrawi and McCallum [18] first introduced the concept of **Multiclass Multilabel Perceptron (MMP)**, modelling label dependencies using structured features and optimising the joint prediction margin across label combinations. Expanding on this framework, Mencía and Furnkranz [38] developed a pairwise multilabel perceptron that efficiently handles large-scale problems by updating only when label rankings are incorrect. Empirical results demonstrate competitive ranking performance, particularly in terms of label ranking loss and precision at top-k, outperforming traditional baselines such as BR and LP.

In summary, algorithm adaptation methods offer a direct and expressive approach to multi-label prediction. When applied appropriately, they can match or exceed the performance of problem transformation techniques, particularly in domains requiring nuanced modelling of label dependencies or where integrated learning architectures are preferred.

2.1.3 Deep Learning for Multi-Label Classification

Modern deep learning architectures naturally support multi-label classification by using **sigmoid activation functions** in the output layer, allowing independent probability estimation for each label. This contrasts with the softmax activation used in single-label classification and enables flexible modelling of label combinations. Deep learning models also offer powerful feature extraction capabilities, learning dense representations of input text that capture both local and long-range dependencies [8, 15].

Different neural network architectures have been adapted for MLC. Convolutional Neural Networks (CNNs) excel at detecting local n-gram patterns, while Recurrent Neural Networks (RNNs), particularly LSTMs and BiLSTMs, capture sequential dependencies across tokens [8, 24]. Multilayer Perceptrons (MLPs) are often used in the output layer, applying sigmoid activation for multi-label prediction [15].

Empirical results support the effectiveness of these models across domains. Johnson et al. [24] use a BiLSTM with attention for clinical text classification, significantly outperforming traditional baselines such as ML-kNN and BR. Chalkidis et al. [8] demonstrate the scalability of deep sigmoid-based models for legal document classification, handling nearly 200 labels and tens of thousands of documents with strong results. Veeranki et al. [59] report that deep models with sigmoid outputs outperform classical baselines like Logistic Regression and Naive Bayes in multi-label clinical settings. Xu et al. [65] show that deep architectures, particularly recurrent and convolutional models, generally outperform traditional methods on large datasets. However, they say that deep learning models demand extensive hyperparameter tuning, large labelled datasets, and GPU acceleration, which can be prohibitive in low-resource environments such as engineering issue logs or specialised domains with limited data.

Overall, while deep neural networks achieve state-of-the-art performance in multi-label tasks with sufficient training data, simpler methods may be more suitable in data-scarce scenarios due to their lower computational and data requirements. Hence, model selection for MLC in industrial environments should weigh performance benefits against computational cost and data availability.

2.1.4 Transformer-Based Models

Transformer-based models [58] have reshaped natural language processing (NLP) by introducing architectures that effectively model long-range dependencies through self-attention mechanisms. Unlike traditional deep learning models such as CNNs or RNNs, which process input sequentially or in local windows, transformers capture global context in a single pass. This capacity is particularly beneficial for multi-label classification, where semantic overlap, long input sequences, and complex label dependencies are common.

A key breakthrough was the introduction of **BERT (Bidirectional Encoder Representations from Transformers)** by Devlin et al. [13]. Pretrained using masked language modelling on large corpora, BERT enables bidirectional contextual encoding that generalises well across tasks. Its suitability for MLC has been confirmed in numerous studies. Chalkidis et al. [8] reported that BERT-based models achieved strong performance in large-

Method	Label Dependency	Scalability	Small Data Friendly	Inter-pretability	Inference Speed	Incremental Learning	Ref.
Problem Transformation							
Binary Relevance (BR)	✗	High	High	High	High	Yes	[21, 4]
One-vs-Rest (OvR)	✗	Moderate	Moderate	High	High	Yes	[57]
Classifier Chains (CC)	Partial (sequential)	Moderate	Moderate	Moderate	Moderate	No	[21, 59]
Label Powerset (LP)	High (joint)	Low	Low	Low	Low	No	[8, 21]
Algorithm Adaptation							
ML-kNN	Partial (local)	Moderate	High	Moderate	Moderate	No	[67]
RankSVM	High (ranking)	Low	Moderate	Low	Low	No	[23]
MMP / Perceptron	High (structured)	Moderate	Moderate	Low	Moderate	No	[18, 38]
Neural Models							
RNN / CNN / BiLSTM / MLP	Partial	Moderate	Low	Low	Moderate	No	[24]
Transformer (BERT)	High (contextual)	Moderate	Low	Low	Low	No	[13, 8]
Transformer (RoBERTa)	High (contextual)	High	Low	Low	Low	No	[68, 31]
Transformer (DistilBERT)	High (contextual)	High	Low	Low	High	No	[68, 50]

Table 2.1: Comparison of multi-label classification methods based on label dependency modelling, scalability, interpretability, and deployment suitability. **Note:** "Partial" = weak or implicit label dependency modelling. Inference speed and scalability are approximate and depend on implementation. "Incremental Learning" indicates support for online updates without retraining from scratch.

scale legal classification, demonstrating their effectiveness in handling tens of thousands of documents and nearly 200 distinct labels. More recently, studies by Schonlau et al. [51] have further confirmed BERT's efficacy, showing it achieved the smallest loss compared to other methods in multi-label classification of open-ended questions.

Building on BERT, **RoBERTa** [34] refines the pretraining process by removing the next sentence prediction objective, applying dynamic masking, and training on more data with larger batches. These improvements enable RoBERTa to learn richer contextual representations. Recent articles provide strong evidence for RoBERTa's enhanced performance in MLC. For instance, Zhang and Xu [68] highlight RoBERTa's stronger performance and its ability to effectively handle challenges like small sample datasets and unbalanced labels in multi-label text classification. Furthermore, in hybrid architectures for automatic multi-label classification, RoBERTa serves as a robust core language model, outperforming baselines like BERT and LSTM in complex setups [31].

For applications requiring low-latency inference or operating under hardware constraints, **DistilBERT** [50] offers a compelling alternative. Created through knowledge distillation, it retains approximately 97% of BERT's performance on the GLUE benchmark while being 40% smaller and 60% faster. Its effectiveness in multi-label tasks is increasingly being documented in research. For example, in competitive settings like the EXIST2024 shared task on sexism categorisation, DistilBERT variants were directly applied and demonstrated strong performance in multi-label classification scenarios [44]. Furthermore, DistilBERT's competitive performance has been observed in practical settings such as Hugging Face pipelines and Kaggle challenges. Veeranki et al. [59] confirmed that fine-tuned BERT and its variants consistently outperform classical baselines like Logistic Regression and Naive Bayes, especially in tasks involving sparse, overlapping label sets.

Despite their empirical success, transformers come with notable challenges. Fine-tuning requires significant computational resources, including GPU memory, and can be sensitive to hyperparameter settings [32]. Moreover, in domains with limited labelled data or highly imbalanced classes, their performance may degrade, and simpler models such as Ridge Classifiers or XGBoost may offer more robust and interpretable alternatives [60].
Nonetheless, transformer-based models represent the current state of the art in multi-label classification. By offering a unified framework for capturing complex language patterns and label interactions, they enable substantial performance gains across diverse domains, including healthcare, legal text, and software engineering. In this work, we comparatively evaluate BERT, RoBERTa, and DistilBERT to identify effective trade-offs between predictive performance and practical constraints in industrial multi-label classification tasks.
Table 2.1 summarises the reviewed multi-label classification methods based on their capabilities and practical considerations for deployment in real-world settings.

2.2 Association

Identifying associations between similar issues is a key task in engineering settings, where knowledge reuse and early recognition of recurring problems can reduce resolution time and improve development processes. The primary challenges in issue association discovery stem from the lack of structured linking mechanisms, meaning issues are typically stored as individual reports without systematic connections to similar or previously resolved ones. Furthermore, the high volume and diversity of reports in engineering databases make manual association impractical. Semantic complexity, where different teams use varying terminologies for similar issues, also hinders simple keyword-based approaches.
To address these challenges, Natural Language Processing (NLP) techniques are applied to automate association discovery by identifying semantic similarities between reports, linking related issues, and recommending relevant cases for engineers. To automate semantic association discovery, the literature proposes a variety of NLP-based strategies, including retrieval-based models, embedding-based approaches, clustering, and topic modelling.

2.2.1 Classical Information Retrieval Methods

Traditional information retrieval techniques have been widely adopted for issue and document similarity tasks. Among these, the **BM25** ranking function has remained a standard due to its effectiveness in handling term frequency and document length variations. The comprehensive survey by Robertson et al. [49] delves into the probabilistic relevance framework, including BM25 and its extensions, highlighting its foundational role and continued relevance in information retrieval applications. Its application to short-text matching and document retrieval tasks has made it a strong baseline in various recommendation scenarios, including issue and bug tracking systems.
BM25 is often favoured for its computational efficiency and interpretability, providing a strong baseline for comparison against more complex, modern approaches in various recommendation tasks. Despite its strengths, BM25 relies primarily on lexical overlap, limiting its effectiveness in cases with synonymy, paraphrasing, or domain-specific terminology. To

address these limitations, more recent methods based on semantic embeddings and deep learning have been proposed.

2.2.2 Semantic Embeddings for Similarity

To capture the deeper contextual meaning of text, semantic embeddings have been increasingly utilised for association discovery. Models such as **Word2Vec [39], GloVe [43], FastText [6]** and, more recently, **Sentence-BERT (SBERT)** [47] are used to encode issue descriptions into dense vector representations. These representations enable similarity to be computed via cosine distance, supporting the recommendation of semantically related issues even in the presence of lexical variation. Semantic embedding-based similarity is especially valuable in engineering settings where semantically related issues may use domain-specific phrasing but share underlying structure.
The work by Reimers and Gurevych [47] introduced SBERT, demonstrating its ability to generate highly effective sentence embeddings for tasks like semantic textual similarity. Beyond general document similarity, studies like Ostendorff et al. [42] have evaluated various document representations, including those derived from methods like FastText, for content-based recommendations, specifically in legal literature, showcasing their applicability to broader document recommendation tasks.
These embedding-based approaches are known for their ability to capture nuanced semantic relationships beyond lexical overlap, making them effective for tasks where synonyms, paraphrases, or related concepts are important for identifying associations.

Method	Semantic Understanding	Scalability	Interpretability	Requires Labels	References
IR-based					
BM25	✗	High	High	✗	[49]
Embedding-based					
TF-IDF + Cosine	✗(lexical)	High	High	✗	[62]
Word2Vec / GloVe + Cosine	✓(shallow)	High	Moderate	✗	[39, 43]
FastText + Cosine	✓✓(subword-aware)	High	Moderate	✗	[6]
SBERT + Cosine	✓✓(contextual)	Moderate	Moderate	✗	[47]
Topic Modelling					
BERTopic / LDA	✓(abstract)	Moderate	High	✗	[20, 2]
Clustering-based					
k-Means / DBSCAN / HDBSCAN	Depends on representation	Moderate	Moderate	✗	[54, 12, 37]

Table 2.2: Comparison of methods used for association discovery based on semantic capabilities, scalability, interpretability, and data requirements. **Note:** ✓= shallow semantics, ✓✓= contextual semantics, ✓✓✓= deep semantics. "Requires Labels" indicates if supervised training data is needed.

2.2.3 Topic Modelling and Clustering

To uncover patterns in large sets of textual data, topic modelling has been applied in the context of issue clustering and exploration for association discovery. **BERTopic**, which combines transformer-based embeddings with class-based TF-IDF, has demonstrated the

ability to form interpretable and coherent topic groups [20]. Such methods can be used to support thematic recommendations by associating new issues with existing topic-based clusters.

While BERTopic builds on deep contextual embeddings, traditional topic modelling techniques also remain valuable for unsupervised issue grouping. **Latent Dirichlet Allocation (LDA)** has been widely applied to categorise and group documents. For instance, Aljedaani et al. [2] utilised LDA for the categorisation of security bug reports in Chromium Projects, demonstrating its utility in organising and identifying themes within software engineering issues. More broadly, a common approach involves combining TF-IDF representations with clustering algorithms such as **k-Means** [54] or **DBSCAN** [12], followed by cosine similarity for grouping similar documents. Furthermore, the effectiveness of TF-IDF based representations combined with cosine similarity for document similarity has been detailed in recent work [62]. For density-based clustering, a state-of-the-art survey on semantic similarity for document clustering using GloVe and density-based algorithms [40] further discusses methods like DBSCAN for document clustering with cosine similarity.

Topic modelling and clustering approaches provide a way to identify underlying themes within a collection of documents, which can then be leveraged to group or associate similar concepts – even if they do not share exact keywords, making them well-suited for domains like engineering where terminology can vary across teams and subsystems.

A structured comparison of association discovery methods is presented in Table 2.2, highlighting their semantic capabilities and suitability for engineering issue retrieval.

2.3 LLMs as Evaluators

While traditional evaluation of NLP models often relies on human annotation or automated metrics (e.g., accuracy, precision, recall), recent research has explored the use of LLMs as surrogate evaluators. These models, such as GPT-4 and Claude, have demonstrated impressive performance in following instructions, making them suitable for tasks like judging the quality of generated content, summarisation, or ranking [33, 61].

In particular, LLMs have been proposed as scalable alternatives to human annotators in settings where manual evaluation is expensive or infeasible. Liu et al. [33] introduce GPTEVAL, a framework for evaluating natural language generation (NLG) outputs using GPT-4 with structured prompts, and show that LLM-based judgments align closely with human preferences in summarisation and dialogue evaluation. Similarly, Wang et al. [61] investigate the fairness and consistency of LLMs as judges and highlight their potential when carefully prompted, while also cautioning against over-reliance in high-stakes or nuanced tasks.

Despite their promise, LLM-based evaluation is still an emerging area with open challenges. These include sensitivity to prompt phrasing, lack of domain-specific knowledge, and potential inconsistencies when evaluating borderline or ambiguous cases [61]. Moreover, most existing studies focus on general-domain tasks; few have rigorously evaluated LLMs as judges in technical or specialised domains, such as engineering or issue tracking.

Motivated by this gap, this thesis incorporates an LLM-based evaluation component as part of the association discovery task. By comparing LLM-generated relevance scores with human expert annotations, this work contributes to the growing body of research on LLMs as evaluators – offering new insights into their reliability and limitations in industrial NLP settings.

2.4 Summary and Research Gap

This chapter reviewed the literature on multi-label classification and issue association discovery. While a wide variety of approaches have been proposed across domains – ranging from classical machine learning to transformer-based models and semantic similarity methods – most are developed and evaluated on benchmark datasets with well-structured, balanced labels. In contrast, real-world issue tracking systems present challenges such as evolving data, noisy input, sparse or missing labels, domain-specific terminology, and overlapping categories.

In the classification domain, classical methods like Binary Relevance and Logistic Regression remain effective for high-dimensional, sparse datasets, especially when computational efficiency and interpretability are priorities. Deep learning models, including CNNs, BiLSTMs, and transformer-based architectures like BERT and RoBERTa, have demonstrated superior performance when sufficient training data and compute resources are available. However, few studies have systematically evaluated these models on noisy, domain-specific engineering data.

Similarly, existing research on semantic association discovery primarily focuses on document similarity or open-domain recommendation. Approaches such as BM25, word embeddings, and topic modelling have shown strong results in general settings but are rarely validated within engineering workflows. Moreover, evaluation is often limited to automated metrics, with limited involvement of domain experts to assess real-world relevance.

Recent advances in LLMs have introduced new opportunities for scalable evaluation. Yet, their reliability in specialised domains like engineering – particularly when compared to expert judgments – remains an open research question.

This thesis addresses these gaps by: (1) systematically evaluating classical and deep learning transformer-based models for multi-label classification on real-world engineering issues; (2) comparing multiple methods for semantic association discovery; and (3) assessing the reliability of LLM-based evaluation by comparing model-generated scores with expert annotations. These contributions offer both empirical evidence and practical guidance for deploying NLP solutions in complex industrial environments.

Chapter 3

Methodology

To systematically analyse and improve issue tracking in an engineering context, this research follows the Cross-Industry Standard Process for Data Mining (CRISP-DM) framework [63], illustrated in Figure 3.1. CRISP-DM provides a structured and iterative approach for data-driven projects, ensuring that business objectives are aligned with data modelling efforts. The methodology consists of six interconnected phases: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment.

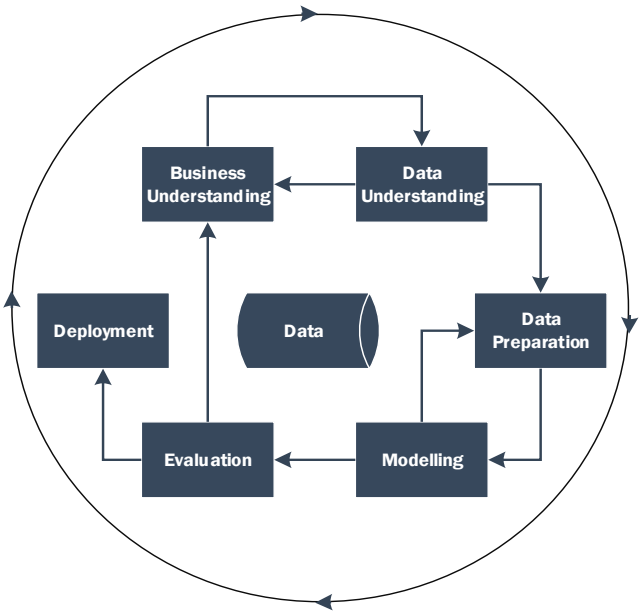


Figure 3.1: Overview of the CRISP-DM framework used to guide the research methodology. The process is cyclical and includes six phases.

1. **Business Understanding:** The first phase focuses on defining project objectives and aligning them with the organizational needs. The issue tracking system contains thousands of reports, making manual classification, association, and resolution slow and

error-prone. To address this, the research applies NLP and ML techniques to automate multi-label issue classification and discover meaningful associations between reports, with the goal of enhancing the efficiency and scalability of issue tracking in engineering workflows.

2. **Data Understanding:** This phase focuses on exploring data from the issue tracking system, assessing its structure, and ensuring data quality. It aims to establish a comprehensive understanding of the dataset's characteristics and domain-specific challenges, such as label sparsity and domain-specific vocabulary, ensuring a solid foundation for subsequent analyses.

3. **Data Preparation:** This phase involves preprocessing and transforming data to ensure consistency and suitability for analysis. Key steps include text cleaning, tokenisation, lemmatisation, and feature extraction to standardise text representations and support effective model training.

4. **Modelling:** This phase focuses on implementing multiple AI/ML approaches tailored to the classification and association tasks. Various techniques are applied to develop models that automate issue classification and identify report associations.

5. **Evaluation:** This phase involves systematically evaluating model performance using task-appropriate evaluation metrics that align with the business objectives of the issue tracking system. The choice of metrics is guided by the specific requirements of each task, ensuring that models prioritise relevant aspects such as recall to enhance classification effectiveness and overall system efficiency.

6. **Deployment:** While full deployment is beyond the scope of this project, this phase provides recommendations for integrating AI models into the organization's issue tracking system. The findings are analysed to ensure practical applicability, and suggestions are made for implementation strategies that enhance issue classification and report association.

This structured approach ensures reproducibility and alignment with business objectives across each CRISP-DM phase, addressing key challenges in issue tracking within complex engineering environments. By systematically applying NLP and AI/ML techniques, this research enhances issue classification and report association. The following chapters will provide a detailed examination of these phases, with a particular focus on data preparation, modelling strategies, and evaluation.

Chapter 4

Classification

Accurate and efficient classification of issue reports is essential for managing technical problems in large-scale engineering projects. This chapter presents the classification component of the research, which focuses on automatically assigning multiple relevant categories to engineering issues using artificial intelligence methods, including machine learning and natural language processing. The following sections outline the problem context, describe the methodology, and present the key results.

4.1 Problem definition

In large-scale engineering projects, issue tracking systems serve as essential tools for documenting, analysing, and resolving technical problems [1]. However, manually categorising these issues into categories remains a significant challenge due to the volume, variability, and complexity of issue descriptions. Misclassifications and inconsistencies in labelling can lead to delays in resolution, inefficient resource allocation, and difficulty in tracking recurring issues across projects [28]. As engineering teams work on diverse systems spanning software, hardware, and design domains, the need for an automated classification system becomes increasingly critical.

The classification of issue reports is inherently complex due to several key challenges. First, multi-label classification is often required since a single issue may belong to multiple categories (e.g., software and verification). However, current classification mechanisms typically assign one label per issue, failing to capture its multi-dimensional nature. A key challenge is that the current database predominantly contains single-label assignments, as the capability to assign multiple labels was only recently introduced. Consequently, the limited availability of multi-labelled data may hinder model training, potentially affecting performance in learning accurate label associations. Additionally, the class distribution is highly imbalanced, with certain categories appearing more frequently than others, making it difficult for models to achieve balanced performance across all issue types.

To address these challenges, this research explores ML and NLP techniques to automate issue classification. The goal is to develop a system that can:

1. Automatically categorise issues, improving consistency and reducing manual effort.

2. Handle multi-label classification, ensuring that issues spanning multiple domains are correctly identified.

By implementing and evaluating AI-driven classification models, this research aims to enhance the efficiency of issue management, ensuring that engineering teams receive faster and more accurate insights into problem categorisation.

4.2 Methodology

This section outlines the methodology developed to address the problem of automated multi-label classification of engineering issues in large-scale projects. The approach integrates data collection, data preprocessing, data cleaning, feature engineering, and evaluation of diverse classification strategies, ranging from classical machine learning to transformer-based deep learning. Given the presence of incomplete label annotations and the multi-label nature of the task, model selection and evaluation were designed to prioritise recall over precision to maximise the likelihood of retrieving relevant categories.
An overview of the full classification methodology pipeline is illustrated in Figure 4.1 and Figure 4.8. The first diagram outlines the business understanding and data preparation steps, while the second depicts the modelling, evaluation, and recommendation phases.

4.2.1 Dataset Construction

The raw dataset was compiled from 31 engineering projects, resulting in an initial corpus of 63,329 issue reports described by 352 attributes. To tailor the dataset to the classification task, only the `Title`, `Description`, and `Category` fields were retained. While the raw dataset contained over 350 attributes, most of these represented metadata generated during or after the issue resolution process (e.g., priority, resolution date, assigned engineer). In contrast, the `Title` and `Description` are available at the very beginning of the issue lifecycle, precisely when classification is most valuable to direct the issue toward the correct team or workflow. Although incorporating additional metadata could potentially improve model performance, it would not reflect the real-world deployment scenario, where classification needs to occur immediately after issue creation. Thus, restricting inputs to `Title` and `Description` not only reduces complexity and noise but also ensures that the model remains practical and applicable to realistic industrial workflows. These data filtering and preprocessing steps are summarised in the first part of the pipeline (Figure 4.1).
Next, records with missing values in any of these fields were excluded, reducing the dataset to 26,998 entries. This step ensures input completeness and prevents introducing bias or errors during model training. Duplicate records – defined as entries with identical `Title` and `Description` – were also removed to avoid overfitting and inflating performance metrics from redundant samples [35].
To mitigate noise and improve model robustness, records with exceptionally long descriptions (above the 99th percentile in word count) were filtered out, as these often contained verbose notes or irrelevant logs that were not suitable for generalisation. This effect can be seen in Figure 4.2, where the distribution of raw description lengths is highly skewed,

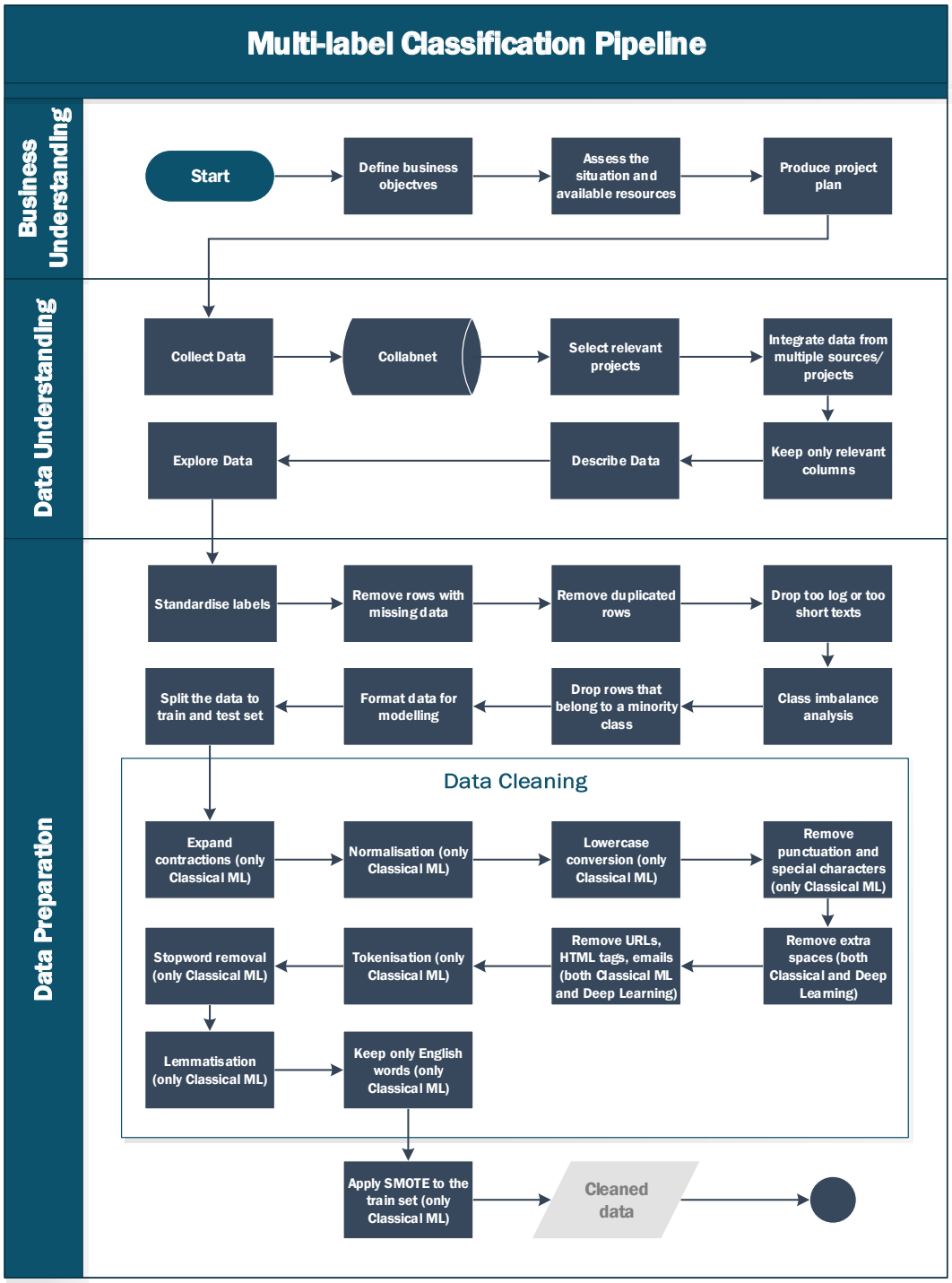


Figure 4.1: Overview of the multi-label classification pipeline developed for this study. The process includes business understanding, data understanding, and a detailed data preparation stage, followed by specific preprocessing pipelines tailored for classical and transformer-based machine learning models.

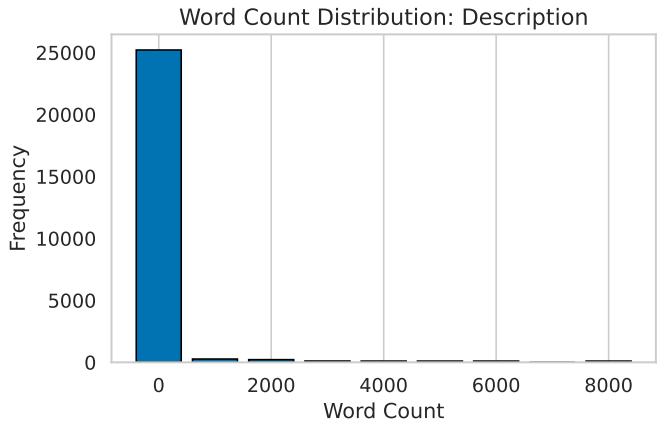


Figure 4.2: Word Count Distribution: Raw Description Field

with a long tail extending beyond 2000 words. After filtering out extreme outliers, the cleaned description length distribution (Figure 4.3) shows a much more manageable range, facilitating generalisation for downstream models.

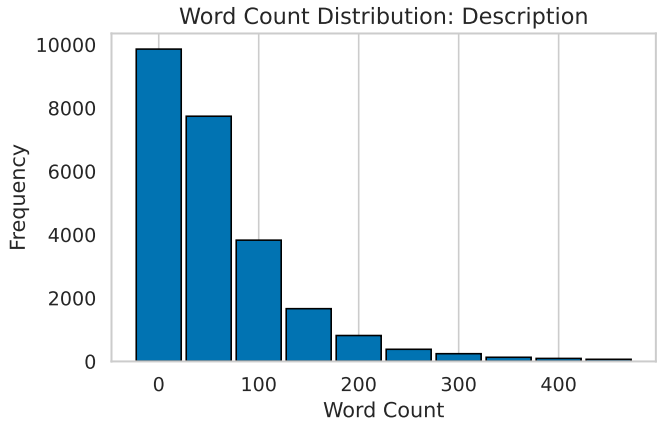


Figure 4.3: Word Count Distribution: Cleaned Descriptions

Additionally, entries in which the combined `Title` and `Description` contained fewer than 20 words (the fifth percentile of the length distribution) were excluded, as such short inputs typically lacked sufficient semantic content for reliable label prediction [1]. Figure 4.4 shows the distribution of combined input lengths, with a vertical red line marking the fifth percentile threshold. These filtering steps collectively refined the dataset to 23,621 high-quality, information-rich issue reports.
To further illustrate the nature of inputs, Figure 4.5 displays the distribution of title lengths, showing that most titles are concise (10–15 words), consistent with expectations for brief

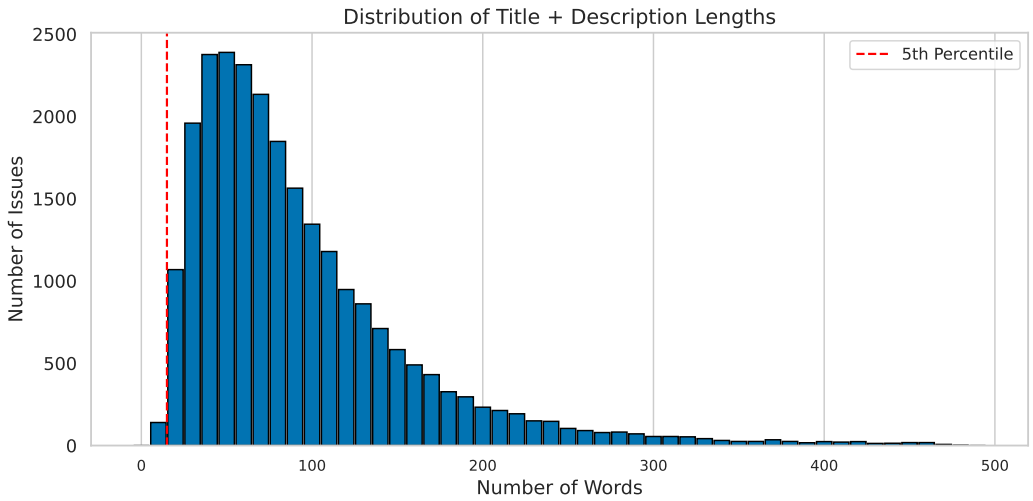


Figure 4.4: Distribution of Title + Description Lengths

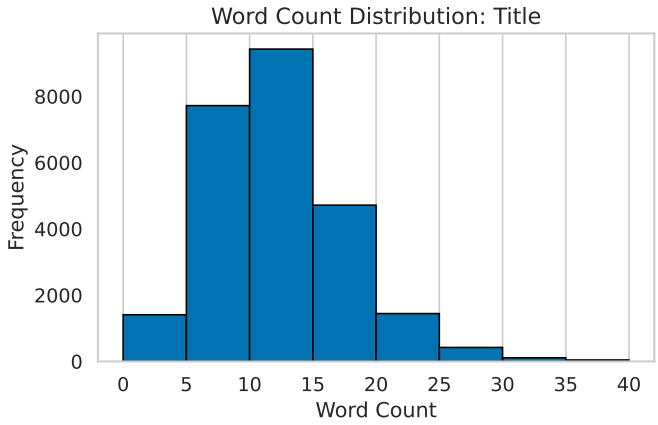


Figure 4.5: Word Count Distribution: Title Field

issue summaries.

Since multi-label classification was only recently introduced in the system, the dataset was significantly skewed toward single-label entries. To ensure reliable learning across categories and avoid sparsity, rare categories occurring fewer than 50 times were removed [57]. The resulting dataset (Figure 4.6 and Figure 4.7) consists of **23,534 issues** spanning **30 categories**, each associated with between **1 and 10 labels**. This step was essential to enable meaningful model learning and fair evaluation across labels. As visible in Figure 4.6, the category distribution remains highly imbalanced, with a few dominant labels such as *ipdesign*, *v&v* and *software* accounting for a large portion of the data. Similarly, Figure 4.7 shows that most multi-label issues are annotated with only two categories, underscoring

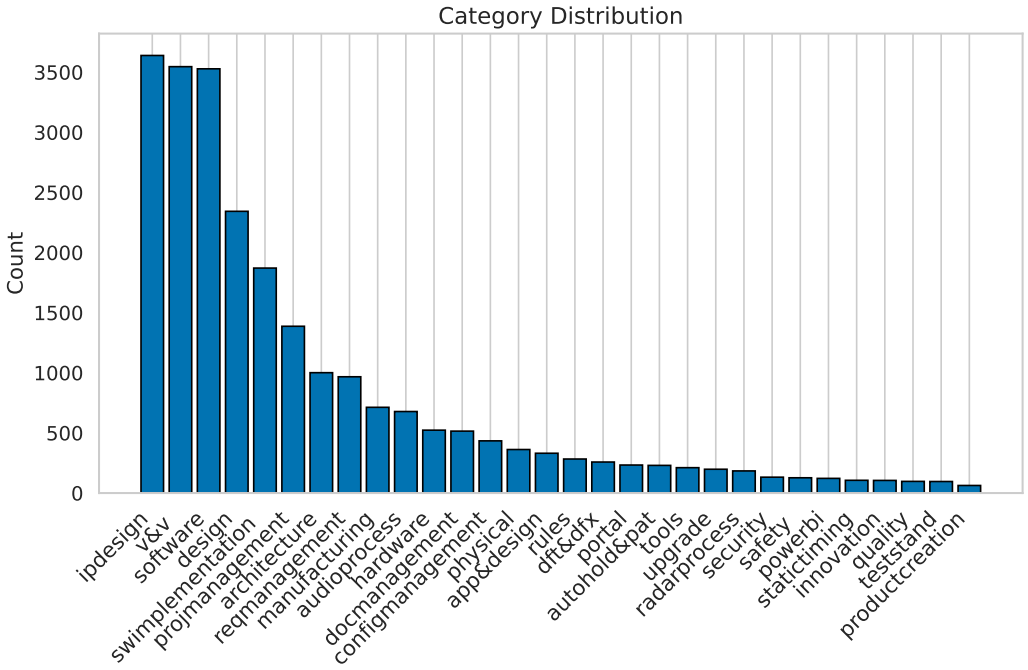


Figure 4.6: Category Distribution Across the Final Dataset

the skew toward low label cardinality. This distribution reflects the relatively recent adoption of multi-label annotation practices and highlights the challenge of learning from limited multi-label examples, further justifying the need for label-aware stratification and balancing techniques during training.

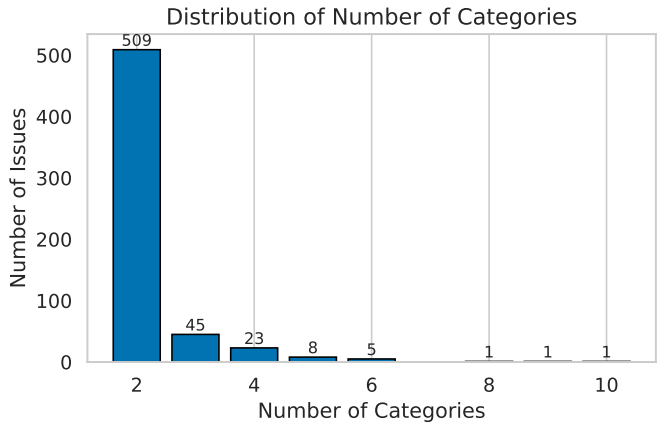


Figure 4.7: Distribution of Number of Categories Assigned per Multi-Label Issue

For downstream experimentation, the dataset was further stratified to support comparative evaluation [5]:

- A single-label dataset of 22,941 records was extracted and stratified into training (80%, 18,352 records) and test (20%, 4,589 records) subsets.
- A multi-label dataset of 543 records was split into training and test sets using label distribution-preserving stratification. Among 1,000 stratified random splits, the one minimizing deviation from the overall label frequency distribution was chosen, producing 475 training and 118 test samples.

The final training and testing datasets, totalling **18,827** and **4,707** issues respectively, form the basis for all classification experiments. These preprocessing steps were crucial for reducing noise, ensuring class representation, and enabling scalable, fair evaluation of classical and transformer-based multi-label classification models.
After stratified splitting, separate preprocessing pipelines were applied to the training and test sets. To address class imbalance in the training set, the Synthetic Minority Oversampling Technique (SMOTE) was applied **only to the training portion** of the dataset. SMOTE creates synthetic examples of minority classes by interpolating between existing samples, improving classifier robustness on underrepresented categories [9]. This step was deliberately applied **after** the train/test split to prevent data leakage, and was used exclusively for classical machine learning models. Transformer-based models rely on alternative mechanisms (e.g., contextual embeddings or class weighting) to manage class imbalance and therefore do not use oversampling.

4.2.2 Data Cleaning

Two separate text cleaning pipelines were applied to the dataset, each tailored to the requirements and assumptions of the downstream models. These preprocessing strategies aimed to ensure data consistency, reduce noise, and preserve semantic content for effective classification [30]. The full set of data cleaning operations, grouped by model type, is visualised in Figure 4.1.

Classical Machine Learning Pipeline

For classical ML models, a more aggressive normalisation approach was adopted to reduce vocabulary size and eliminate irrelevant variability. Such models typically operate on sparse, frequency-based representations (e.g., TF-IDF), which are highly sensitive to inconsistent formatting or rare tokens. Therefore, standardisation was critical to improve model generalizability and reduce overfitting [30].
The following steps were applied to the combined `Title` and `Description` fields:

- **Contraction Expansion**: Common English contractions (e.g., *can't* → *cannot*) were expanded using the `contractions` library.
- **Unicode Normalization**: Accented or non-ASCII characters were normalized via `unicodedata` to their ASCII equivalents.

- **Lowercasing**: All text was converted to lowercase to reduce token sparsity.
- **Punctuation Removal**: Special characters and punctuation were removed using regular expressions, except hyphens and alphanumeric terms.
- **Whitespace Normalisation**: Tabs, newlines, and redundant spaces were standardised.
- **Tokenization**: Text was split into individual words using NLTK's `word_tokenize`.
- **Stopword Removal**: Standard English stopwords were removed to retain only informative terms.
- **Lemmatisation**: Words were reduced to their base form using WordNet lemmatiser to unify morphological variants.
- **Noise Removal**: URLs, HTML tags, and email addresses were removed with regex filters.
- **Dictionary Filtering**: Non-alphabetic tokens and out-of-vocabulary words were filtered.

This multi-stage cleaning process standardised the input text, reduced dimensionality, and improved the suitability of the data for classical vectorisation techniques.

Transformer-Friendly Pipeline

In contrast, transformer-based models such as BERT rely on pre-trained tokenisers and contextual embeddings that are sensitive to linguistic structure, punctuation, and word order. As such, the cleaning process was intentionally conservative in order in order to preserve the syntactic and semantic richness of the input [13].
The following minimal steps were applied:

- **HTML Tag Removal**: Embedded HTML content was removed.
- **URL and Email Removal**: Links and email addresses were removed to avoid irrelevant information.
- **Whitespace Normalization**: Extra spaces and formatting inconsistencies were standardized.

This lighter cleaning approach aimed to retain context-relevant structures and punctuation that transformers use to infer meaning through attention mechanisms. No lowercasing or lemmatisation was applied, as transformer tokenisers are trained on case-sensitive, subword-based vocabularies.

4.2.3 Feature Engineering

For the multi-label text classification task, the combined `Title` and `Description` fields were transformed into numerical representations suitable for machine learning models. A variety of vectorisation techniques were evaluated, including both sparse and dense representations. These methods were selected based on their widespread use in text classification tasks, compatibility with classical models, and their ability to capture different linguistic features. The final representation techniques covered both frequency-based and

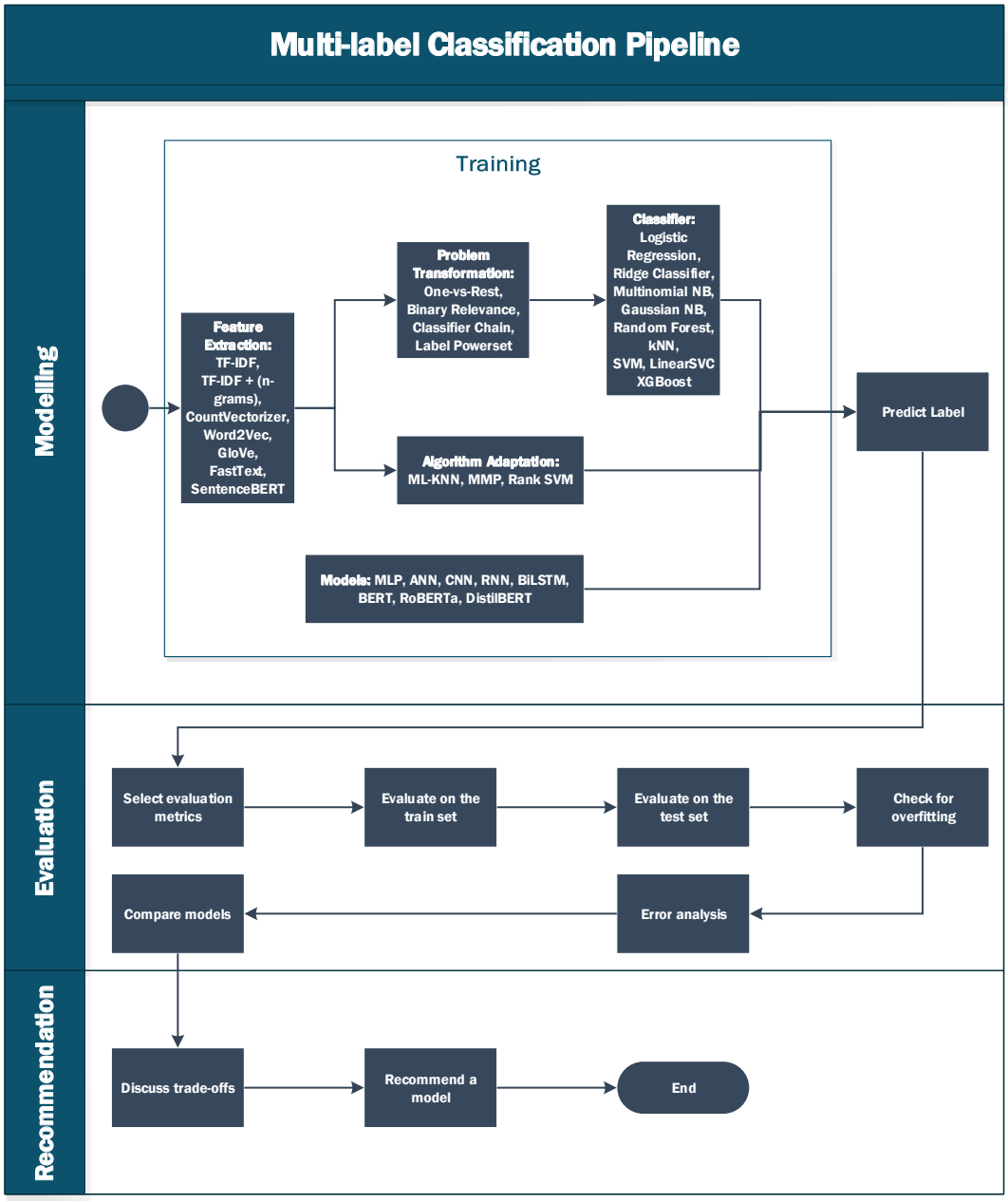


Figure 4.8: Overview of the modelling, evaluation, and model recommendation phases of the multi-label classification pipeline. This diagram illustrates the progression from feature extraction, problem transformation, algorithm adaptation to model training and evaluation. A range of classical and deep learning models, including transformer-based architectures, are assessed. The final step involves model comparison and recommendation.

embedding-based approaches. The feature engineering step, including vectorisation techniques explored, is shown in Figure 4.8.

Frequency-Based Representations

Frequency-based methods represent text using counts or weighted frequencies of individual words [55, 30]. In this study, two such techniques were employed: the **Count Vectorizer (Bag-of-Words)**, which transforms text into sparse vectors by counting raw word occurrences [55], and **Term Frequency–Inverse Document Frequency (TF-IDF)**, which assigns weights to words based on their importance across the corpus [48]. Both unigrams and bigrams were included in the TF-IDF n-gram variant to capture limited contextual information without significantly increasing dimensionality. These sparse representations are particularly well suited for structured and domain-specific texts [30].

Embedding-Based Representations

Embedding-based methods generate dense, continuous vector representations that capture semantic relationships between words or sentences [39]. In this study, several embedding techniques were evaluated: **Word2Vec [39], FastText [6], GloVe [43], and Sentence-BERT (SBERT) [47]**. These methods are widely used in modern NLP applications and provide pre-trained or trainable embeddings that can be integrated into machine learning pipelines [30]. Unlike frequency-based approaches, embeddings encode contextual or distributional meaning, which can improve performance on tasks involving more abstract or nuanced language [47].
Table 4.1 summarises all vectorisation techniques considered in this study.

4.2.4 Classification Strategies

To effectively address the multi-label classification problem, a diverse set of modelling strategies was explored. These were chosen to balance interpretability, scalability, and performance under the constraints of noisy, domain-specific textual input. Three main groups of approaches were considered: problem transformation methods, algorithm adaptation methods, and various base learners ranging from classical models to transformer-based deep learning. All classification strategies are visualised in the modelling phase of Figure 4.8.

Problem Transformation Approaches

Since most conventional classifiers are inherently designed for single-label classification tasks, the multi-label nature of the problem requires transformation strategies to adapt the dataset and modelling pipeline [21]. Four widely used problem transformation methods were selected based on their prevalence in the literature, conceptual variety, and compatibility with classical machine learning models.

- **One-vs-Rest (OvR)** [5]: This method trains a separate binary classifier for each label. It assumes label independence and allows the use of any base learner. Its interpretability and simplicity make it a strong baseline in multi-label classification tasks.

Method	Short Description	Vector Size	Reference
TF-IDF	Assigns weights to terms based on their frequency in a document and rarity across the corpus. Includes unigrams and bigrams.	Variable	[48]
Count Vectorizer	Converts text into sparse vectors by counting raw word frequencies. Effective for short and structured technical text.	Variable	[55]
Word2Vec	Learns distributed word representations using a shallow neural network trained on context prediction (Google News).	300	[39]
FastText	Extends Word2Vec by including subword information, enabling better handling of rare and out-of-vocabulary words.	300	[6]
GloVe	Generates word vectors using global word co-occurrence statistics. Trained on Wikipedia and Gigaword.	300	[43]
SBERT	Produces contextual sentence embeddings using a BERT-based Siamese architecture. Optimised for semantic similarity tasks.	768	[47]

Table 4.1: Summary of the text vectorisation methods used in the multi-label classification task. The table describes approaches, ranging from sparse representations like TF-IDF and Count Vectorizer to dense embeddings such as Word2Vec, FastText, GloVe, and SBERT. Each method differs in dimensionality and semantic richness, contributing distinct strengths to downstream model performance.

It originates from multi-class classification but is commonly adapted for multi-label tasks.

- **Binary Relevance (BR)** [5]: Each label is treated as an independent binary classification problem. While it does not model any inter-label correlations, it is computationally efficient and often surprisingly effective for problems with weak label dependencies. Unlike OvR, BR is specifically designed for multi-label classification and handles multiple binary predictions per instance by default.
- **Classifier Chains (CC)** [5]: Unlike OvR and BR, this method accounts for label dependencies by passing predicted labels as input features to subsequent classifiers in a chain. This approach was chosen to explore whether modelling label interdependence would improve predictive accuracy.
- **Label Powerset (LP)** [5]: This method transforms the multi-label problem into a multi-class one by treating each unique label combination as a single class. Although powerful in theory, it often struggles with label sparsity and scalability when many label combinations exist.

These techniques provide a modular and comparative framework for reusing classical models in a multi-label setting, allowing systematic evaluation across different assumptions of label independence, dependency, and class distribution [21]. They were chosen to span a range of modelling strategies – from fully independent (Binary Relevance) to fully joint

(Label Powerset) – to better understand which approaches align best with the structure of the task.

Algorithm Adaptation Techniques

In addition to transformation-based approaches, several algorithm adaptation techniques were evaluated. Unlike transformation methods, which reformulate the task to suit standard classifiers, these methods are specifically designed to handle multi-label learning natively [5]. They were selected based on their theoretical relevance to the task, particularly their ability to model label correlations or rank label relevance directly.

- **ML-KNN** [67]: An extension of the standard k-nearest neighbours algorithm, ML-KNN estimates the probability of each label being relevant based on label frequencies among the nearest neighbours. It is well-suited for small datasets and offers an interpretable, instance-based approach. However, it can struggle with high-dimensional or noisy data due to its reliance on local similarity.
- **RankSVM** [23]: This method frames multi-label classification as a label ranking problem. It learns a pairwise preference function over label pairs using a support vector machine. RankSVM was included to explore whether ranking-based formulations could better capture partial relevance among labels, which is often useful in noisy or weakly labelled environments.
- **Multiclass Multilabel Perceptron (MMP)** [18]: MMP optimises a margin-based objective function to predict a ranked list of labels per instance. It integrates collective inference to exploit label dependencies, aiming to balance precision and recall in the predicted label sets. The method was included to evaluate structured prediction capabilities, particularly in scenarios with overlapping label semantics.

These methods were selected to evaluate whether algorithmic adaptations that model label correlations or relevance rankings directly offer advantages over transformation-based approaches in this context.

Base Learners: Classical Models, Deep Learning Models and Transformers

Each transformation strategy was paired with a variety of base classifiers to explore differences in generalisation ability, computational efficiency, and interpretability across models:

Classical Models. A diverse set of classical machine learning algorithms was evaluated for their efficiency, interpretability, and established effectiveness in text classification tasks. These included linear models, tree-based ensembles, probabilistic approaches, and instance-based methods:

- **Logistic Regression:** A linear model that estimates the probability of each label using the logistic function [29].
- **Ridge Classifier:** A regularized linear classifier that applies L_2 penalty to prevent overfitting [5].

- **Support Vector Machine (LinearSVC):** A margin-based classifier that finds an optimal separating hyperplane in high-dimensional space [69].
- **Random Forest:** An ensemble of decision trees that performs classification through majority voting across randomly constructed trees [7].
- **XGBoost:** A gradient boosting framework that builds decision trees sequentially to correct previous errors and minimise classification loss [10].
- **Naive Bayes (Multinomial and Gaussian):** Probabilistic classifiers based on Bayes' theorem with strong (naive) independence assumptions among features [21].
- **k-Nearest Neighbours (kNN):** A non-parametric method that assigns labels based on the majority class among the k most similar training instances [21].

These classical classifiers were selected due to their extensive application and evaluation in multi-label learning literature. As noted in [21, 57], kNN and Naive Bayes remain popular for their simplicity and resilience to data imbalance; SVMs offer strong margin-based generalization despite scalability issues; logistic regression and its regularized variants (e.g., ridge) are valued for interpretability and robustness; ensemble methods like Random Forest and boosting approaches like XGBoost are widely adopted for their ability to model complex label dependencies and improve predictive performance.

Neural Networks. Both shallow and deep neural architectures were explored to assess their ability to model non-linear patterns and hierarchical representations in text. These included feed-forward and sequence-based models:

- **Artificial Neural Networks (ANN):** General feed-forward networks composed of one or more hidden layers with non-linear activation functions [70].
- **Multi-layer Perceptrons (MLP):** A specific class of ANN with fully connected layers, typically used as a baseline for non-linear classification tasks [17].
- **Convolutional Neural Networks (CNN):** Networks that apply convolutional filters over input sequences to capture local n-gram patterns and spatial hierarchies [64].
- **Recurrent Neural Networks (RNN):** Sequence models that process input tokens sequentially, maintaining hidden states to capture temporal or contextual dependencies [53].

Transformer-based Models. Pretrained transformer architectures were incorporated to evaluate their ability to capture deep contextual semantics and model complex label dependencies in text. The selected models were fine-tuned locally for the multi-label classification task:

- **BERT (Bidirectional Encoder Representations from Transformers):** A deep transformer model that uses bidirectional self-attention to learn context-aware word representations [13].
- **RoBERTa (Robustly Optimised BERT Pretraining Approach):** An improved variant of BERT trained with more data and optimised hyperparameters, yielding stronger performance across NLP tasks [34].

- **DistilBERT:** A lightweight, distilled version of BERT that retains much of its performance while significantly reducing model size and inference time [50].

4.2.5 Evaluation Metrics

To rigorously assess the performance of the classification models, a comprehensive set of evaluation metrics was employed. Given the multi-label nature of the task, where each issue report may belong to multiple categories, traditional single-label metrics are insufficient. Instead, this study adopts metrics that account for partial correctness, label ranking, and prediction coverage. Furthermore, given the problem's nature, the business case and the existence of missing labels in the training set, priority was given to recall over precision to minimise the risk of under-predicting valid categories. The following subsections describe each metric in detail, including its mathematical formulation and interpretive value.

Correct Label Count (Relaxed Match)
This metric counts the total number of correctly predicted labels across all instances. For each instance, it computes how many of the ground truth labels are present in the predicted label set. The computation logic is consistent across both single-label and multi-label cases.

$$\text{CorrectLabelCount} = \sum_{i=1}^{N} |\hat{y}_i \cap y_i| \tag{4.1}$$

where N is the total number of instances (single-label or multi-label), y_i is the set of ground truth labels for instance i, $\hat{y}_i$ is the set of predicted labels for instance i, $|\hat{y}_i \cap y_i|$ is the number of correctly predicted labels for instance i.
Single-label case: Each y_i contains exactly one label. The metric counts how many of these labels are present in the predicted sets $\hat{y}_i$. This relaxed version of accuracy considers a prediction correct if the model includes the correct ground truth label among its predictions, even if additional labels are also predicted.
Multi-label case: Each y_i contains multiple labels. For each instance, the metric counts how many of the ground truth labels are present in the predicted label set. The final score is the total number of correctly predicted labels across all multi-label instances.
This unified formulation ensures consistency in evaluation and allows for direct comparison between model performance on different subsets of the data.

Recall@k
Recall@k evaluates the proportion of true labels that appear within the top-k predicted labels for each instance. It is particularly relevant in practical settings where only the top few predictions are presented to users.

$$\text{Recall@k} = \frac{1}{N} \sum_{i=1}^{N} \frac{|\text{Top-}k(\hat{y}_i) \cap y_i|}{|y_i|} = \frac{TP}{TP+FN} \tag{4.2}$$

where Top-$k(\hat{y}_i)$ denotes the top-k predicted labels for instance i, TP (True Positives): Number of correctly predicted positive labels,FN (False Negatives): Number of actual positive labels that were not predicted.
A higher Recall@k indicates that the model is effective at prioritising relevant labels near the top of its predictions.

F1-Score (Macro-Averaged)
The F1-score is the harmonic mean of precision and recall. In multi-label settings, the macro-averaged F1-score is computed by averaging the F1-scores across all labels, treating each label equally regardless of its frequency.

$$\text{F1}_{\text{macro}} = \frac{1}{L}\sum_{j=1}^{L}\frac{2\cdot\text{Precision}_j\cdot\text{Recall}_j}{\text{Precision}_j+\text{Recall}_j} \tag{4.3}$$

where L is the number of labels.

Mean Proportion of Properly Predicted Labels (MPP)
MPP measures the proportion of true labels that are correctly predicted for each instance and then averages this proportion across all instances.

$$\text{MPP} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{y}_i\cap y_i|}{|y_i|} \tag{4.4}$$

Mean Average Precision (MAP)
MAP evaluates the quality of the label ranking by measuring how well the model ranks relevant labels above irrelevant ones. It is computed as the average precision across all relevant labels for each instance.

$$\text{MAP} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|y_i|}\sum_{j\in y_i}\frac{|\{k\in y_i : \text{rank}_i(k)\le\text{rank}_i(j)\}|}{\text{rank}_i(j)} \tag{4.5}$$

where $\text{rank}_i(j)$ is the rank of label j in the predicted list for instance i.

Coverage Error
Coverage error measures how many top-ranked labels need to be considered to include all the true labels for an instance. It reflects the depth of prediction required to fully capture the ground truth.

$$\text{Coverage Error} = \frac{1}{N}\sum_{i=1}^{N}\left(\max_{j\in y_i}\text{rank}_i(j)\right) \tag{4.6}$$

Lower coverage error values are desirable, indicating that fewer predictions are needed to cover all correct labels.

Vectorizer	Transformation	Classifier	Single label Correct (4589)	Multi-label Correct (270)	Total Correct (4859)
		BERT	4240 (92.39)	212 (78.52)	4452 (91.62)
		RoBERTa	4205 (91.63)	216 (80.00)	4421 (90.99)
		DistilBERT	**4260 (92.83)**	**221 (81.85)**	**4481 (92.22)**
TF-IDF	One-vs-Rest	Logistic Regression	3991 (86.97)	179 (66.30)	4170 (85.82)
TF-IDF	Label Powerset	Logistic Regression	3906 (85.12)	**212 (78.52)**	4118 (84.75)
TF-IDF (n-grams)	One-vs-Rest	Logistic Regression	4012 (87.43)	172 (63.70)	4184 (86.11)
Count Vectorizer	One-vs-Rest	Multinomial NB	4014 (87.47)	166 (61.48)	4180 (86.03)
TF-IDF	One-vs-Rest	Multinomial NB	3965 (86.40)	162 (60.00)	4127 (84.94)
TF-IDF (n-grams)	One-vs-Rest	Multinomial NB	3972 (86.55)	168 (62.22)	4140 (85.20)

Table 4.2: Correctly predicted single-label and multi-label issues for various classification pipelines. DistilBERT achieved the highest overall accuracy, while classical models also demonstrated competitive performance. Notably, the Label Powerset approach with Logistic Regression achieved multi-label accuracy comparable to BERT, highlighting its effectiveness despite lower overall accuracy.

Hamming Loss
Hamming loss quantifies the fraction of labels that are incorrectly predicted, either as false positives or false negatives, normalised over all labels and instances.

$$\text{Hamming Loss} = \frac{1}{N \cdot L} \sum_{i=1}^{N} \sum_{j=1}^{L} \delta(y_{ij} \neq \hat{y}_{ij}) \tag{4.7}$$

where y_{ij} and $\hat{y}_{ij}$ are the true and predicted binary values for label j of instance i.

4.3 Results

This section presents the empirical results of the multi-label classification experiments. The performance of the proposed classification pipelines was systematically evaluated using the experimental setup described in the previous section. Multiple combinations of vectorisation methods, transformation strategies, and classifiers were tested to identify the most effective configurations for multi-label issue categorization. As established before, the dataset is inherently incomplete – many relevant labels are missing from the ground truth, which substantially limits the reliability of precision-based metrics. Therefore, Recall@5 and Recall@7 are emphasised throughout this section as primary indicators of model performance. Table 4.3 reports Recall@k scores (k = 1, 3, 5, 7, 10) for all evaluated models. For completeness, additional metrics such as total correct predictions (Table 4.2) and ranking/error-based scores (Table 4.4) are also presented and discussed in the relevant subsections.

4.3.1 Transformer-Based Models

Transformer-based models significantly outperformed traditional methods across all major metrics, with particularly strong gains in **Recall@5** and **Recall@7**. Among the three

Vectorizer	Transformation	Classifier	Recall@1	Recall@3	Recall@5	Recall@7	Recall@10
		BERT	0.63	0.86	0.92	0.96	0.98
		RoBERTa	0.60	0.83	0.91	0.94	0.97
		DistilBERT	0.63	0.86	**0.93**	**0.96**	0.98
TF-IDF	One-vs-Rest	Logistic Regression	0.47	0.76	0.86	0.91	0.96
TF-IDF	Label Powerset	Logistic Regression	0.44	0.73	0.85	0.90	0.95
TF-IDF (n-grams)	One-vs-Rest	Logistic Regression	0.48	0.77	**0.87**	**0.93**	0.97
Count Vectorizer	One-vs-Rest	Multinomial NB	0.47	0.76	0.86	0.92	0.96
TF-IDF	One-vs-Rest	Multinomial NB	0.44	0.73	0.86	0.92	0.96
TF-IDF (n-grams)	One-vs-Rest	Multinomial NB	0.44	0.74	0.86	0.91	0.96

Table 4.3: Recall@k scores (for k = 1, 3, 5, 7, 10) across classification pipelines. DistilBERT achieved the highest Recall@5 (0.93) and Recall@7 (0.96), confirming its strength in ranking relevant labels. Among classical models, TF-IDF with Logistic Regression and n-gram features performed competitively, underscoring their effectiveness despite lower complexity.

Vectorizer	Transformation	Classifier	MPP	MAP	Coverage Error	F1-Score	Hamming Loss
		BERT	0.19	0.76	2.22	0.31	0.14
		RoBERTa	0.19	0.74	2.37	0.31	0.14
		DistilBERT	0.19	0.76	**2.21**	0.31	0.14
TF-IDF	One-vs-Rest	Logistic Regression	0.18	0.64	3.07	0.30	0.14
TF-IDF	Label Powerset	Logistic Regression	0.17	0.62	3.21	0.29	0.14
TF-IDF (n-grams)	One-vs-Rest	Logistic Regression	0.18	0.65	**2.99**	0.29	0.14
Count Vectorizer	One-vs-Rest	Multinomial NB	0.18	0.64	2.98	0.29	0.14
TF-IDF	One-vs-Rest	Multinomial NB	0.18	0.61	3.17	0.29	0.14
TF-IDF (n-grams)	One-vs-Rest	Multinomial NB	0.17	0.62	3.11	0.29	0.14

Table 4.4: Performance of classification pipelines on additional evaluation metrics, including Mean Average Precision (MAP), Coverage Error, F1-Score, and Hamming Loss.

transformer architectures evaluated – BERT, RoBERTa, and DistilBERT – **DistilBERT**, despite being a compressed version of BERT, achieved **the highest** Recall@5 (0.93) and Recall@7 (0.96), outperforming both BERT and RoBERTa (see Table 4.3). These results indicate that DistilBERT is not only computationally more efficient than the other two methods but also highly effective for multi-label classification in this domain. **BERT** followed closely with Recall@5 = 0.92 and Recall@7 = 0.96, while **RoBERTa** trailed slightly (Recall@5 = 0.91; Recall@7 = 0.94).

Beyond Recall@5 and @7, DistilBERT also achieved a **Recall@1** of 0.63, **Recall@3** of 0.86, and **Recall@10** of 0.98 – demonstrating that a large proportion of relevant labels were retrieved early in the ranked prediction list. BERT and RoBERTa produced comparable Recall@10 scores of 0.98 and 0.97, respectively. The ability of these models to rank relevant labels near the top is critical, as selecting too few labels would miss relevant associations, while selecting too many could introduce irrelevant noise, both undesirable in operational settings.

With regard to **accuracy on labelled examples**, DistilBERT correctly predicted 221 out of 270 multi-label examples (81.85%) and achieved a single-label classification accuracy of 4260/4589 (92.83%). This results in a **combined total accuracy of 4481/4859 (92.22%)**, the highest among all evaluated models. In comparison, BERT and RoBERTa attained total correctness rates of 91.62% and 90.99%, respectively, with slightly fewer multi-label matches 212/270 and 216/270 (see Table 4.2).
In terms of **ranking quality** both DistilBERT and BERT achieved a **Mean Average Precision (MAP)** score of 0.76, whereas RoBERTa performed slightly lower with a score of 0.74 (Table 4.4). These high MAP values confirm that the most relevant labels tend to appear earlier in the prediction list, aligning well with the intended use of the system.
Though **F1-scores** across all three models remained modest (0.31), this was expected given the label incompleteness in the training data, which penalises models for predicting labels that may be correct but are unobserved in the ground truth. **Hamming Loss**, which measures the fraction of misclassified labels (both false positives and false negatives) over the total number of labels, provides a view of prediction quality across all labels. Lower values here (DistilBERT: 0.1376; BERT: 0.1380; RoBERTa: 0.1385) suggest that the models make relatively few errors per label, even if not all correct labels are captured. Additionally, **Coverage Error** – which reflects the number of top-ranked labels needed to cover all true labels – was also lowest for DistilBERT (2.21), closely followed by BERT (2.22), and higher for RoBERTa (2.37). This further supports the ranking efficiency of DistilBERT in covering the full set of ground-truth labels with minimal over-prediction.
Taken together, these results underscore the superior performance of transformer-based models, particularly DistilBERT, in handling sparse and incomplete multi-label annotations. Their strong showing across both ranking-sensitive and label-wise metrics affirms their suitability for real-world deployment engineering environments.

4.3.2 Classical Machine Learning

In contrast to the high-performing transformer models, classical machine learning methods based on sparse lexical representations, such as TF-IDF and CountVectorizer, demonstrated lower effectiveness across most evaluation metrics. Nevertheless, some configurations produced competitive scores, prompting further examination due to their computational simplicity and interpretability.
Among the classical pipelines, the best-performing approach in terms of recall was the **Logistic Regression classifier trained on TF-IDF with n-gram features using a One-vs-Rest transformation**. This model achieved a **Recall@5** of 0.87 and **Recall@7** of 0.93, coming relatively close to DistilBERT (Recall@5 = 0.93; Recall@7 = 0.96). Other recall scores for this model included Recall@1 = 0.48, Recall@3 = 0.77, and Recall@10 = 0.97 (see Table 4.3), indicating reasonable performance in early label retrieval. As it can be seen in Table 4.2, this model predicted 172 out of 270 multi-label examples (63.70%), and had a single-label accuracy of 4012/4589 (87.43%), leading to a **total accuracy of 4184/4859 (86.11%)**. The **Coverage Error** was 2.99 (Table 4.4)- nearly one full label higher than DistilBERT (2.22), which suggests the model needed to predict more labels on average to cover the full ground truth. The **MAP** score was 0.65 – higher than all other classical

alternatives but still behind transformers – and the **F1-score** was 0.29. The **Hamming Loss** of 0.14 indicates a modest rate of label-level prediction errors, where the model makes some false positive or false negative predictions per label.
Another strong classical baseline was the **Multinomial Naive Bayes model with CountVectorizer features and a One-vs-Rest strategy**, which achieved **Recall@5** = 0.86, **Recall@7** = 0.92, and **Coverage Error** = 2.98. It correctly predicted 166 multi-label examples (61.48%) and 4014 (87.47%) single-label samples – just two more than the TF-IDF + Logistic Regression model (4012), which led to a **total accuracy of 4180/4859 (86.03%)**. Moreover, it matched the Logistic Regression model in both **F1-score** (0.29) and **Hamming Loss** (0.14).
A noteworthy observation was the **Label Powerset on combination with Logistic Regression** pipeline, which achieved **212 out of 270 multi-label predictions correct (78.52%)** – equal to BERT (see Table 4.2) – despite its lower Recall@K scores (Recall@5 = 0.85, Recall@7 = 0.90) and higher Coverage Error (3.21). This result suggests that, although Label Powerset models may underperform in ranking-based metrics, they are capable of predicting label combinations that align well with the annotated data.
Additional classical configurations such as Binary Relevance and One-vs-Rest with Logistic Regression or Naive Bayes yielded slightly lower performance, as it can be seen in Table 4.4, with F1-scores of 0.29 and MAP consistently below 0.65. These models also had higher Coverage Errors (above 3.00) and marginally worse Hamming Loss scores, confirming the trade-off between simplicity and retrieval quality.
In summary, classical approaches using TF-IDF or CountVectorizer features, combined with Logistic Regression or Naive Bayes, offer efficient and interpretable baselines, particularly in low-resource environments. While their performance in ranking-based metrics and early recall is generally below that of transformer models, the differences are often moderate rather than critical. Some configurations – such as Label Powerset with Logistic Regression – show notable alignment with the annotated label sets, suggesting potential value in scenarios where model simplicity, speed, and transparency are prioritised. These methods remain relevant in practical settings, especially when resource constraints, ease of retraining, or explainability are important considerations.

4.3.3 Additional Models

In addition to the configurations discussed in detail above, several other modelling strategies were explored during the experimental phase. These included classical machine learning models such as Support Vector Machines (LinearSVC), Ridge Classifier, Random Forest, XGBoost, and k-Nearest Neighbours (kNN). While widely used in text classification, these models consistently underperformed across key evaluation metrics such as Recall@5, Recall@7 and Coverage Error, and did not surpass the performance of simpler baselines like Logistic Regression and Naive Bayes.
Algorithm adaptation techniques – including ML-KNN, RankSVM, and MMP – were also evaluated but yielded lower predictive performance in this setting. Similarly, deep learning architectures such as Multi-layer Perceptrons (MLP), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN) failed to outperform the best classical or

transformer-based models. Dense embedding representations – Word2Vec, GloVe, FastText – were tested in combination with these models, but did not lead to improvements over sparse lexical features.
These approaches are not discussed in further detail here due to their lower effectiveness on this dataset. However, full evaluation results for all models and configurations are available in Appendix A.

4.4 Discussion

This study evaluated a diverse set of multi-label classification pipelines to assess their effectiveness in categorising engineering issue description within a complex industrial setting. The industrial setting is characterised by incomplete annotations, class imbalance, and significant semantic overlap between categories. The evaluated methods spanned from simple frequency-based classifiers to advanced transformer-based architectures, offering a comprehensive view of their respective strengths, limitations, and trade-offs.

4.4.1 Model Comparison and Performance Insights

Transformer-based models (BERT, RoBERTa, DistilBERT) demonstrated strong and consistent performance across both single-label and multi-label cases. They effectively disambiguated overlapping technical categories and handled incomplete annotations with robustness. This was reflected in their high Recall@K values, low Hamming Loss and Coverage Error, and strong MAP scores. These outcomes align with findings in recent literature, which report that transformer architectures often outperform classical models on imbalanced and semantically complex datasets [8]. The present study confirms this trend in the context of industrial issue tracking, where label sparsity, semantic overlap, and structured input formats are common.
Among the three transformer models evaluated, **DistilBERT** consistently outperformed both BERT and RoBERTa across most evaluation metrics, despite being a smaller and compressed variant of BERT. DistilBERT achieved the best ranking performance with a **Recall@5 of 0.93**, **Recall@7 of 0.96**, and a **Coverage Error of 2.21**. These results indicate that DistilBERT successfully captures the semantic nuances of engineering-related language and retrieves a high proportion of relevant labels early in the ranked prediction list – a critical capability in tasks with incomplete ground truth.
Several factors may explain DistilBERT's superior performance. First, its smaller size (fewer layers and parameters) makes it inherently **less prone to overfitting**, which is advantageous in low-resource settings or when the training data contains missing or noisy labels. In such cases, larger models like BERT may overfit to noise, while DistilBERT generalises more effectively. Second, DistilBERT typically **converges faster and more stably** during fine-tuning. When training time or compute budgets are limited, this allows for more effective optimisation compared to larger models that may require additional training or regularisation to perform well. Third, DistilBERT benefits from the **knowledge distillation process**, which transfers smoothed decision boundaries from its larger teacher model. This often improves generalisation, especially in domain-specific tasks with structured inputs.

In this study, the combined `Title + Description` inputs were relatively short and technical, likely not requiring the full capacity of BERT or RoBERTa. DistilBERT's distilled architecture was thus better aligned with the task's complexity and constraints.
Taken together, these factors help explain why DistilBERT achieved the strongest performance across both ranking-based and label-wise metrics. Its balance of semantic precision, computational efficiency, and robustness makes it particularly well-suited for real-world engineering issue-tracking scenarios, where scalability, noise tolerance, and deployment feasibility are essential.
In contrast, classical pipelines – particularly **TF-IDF (n-grams) + One-vs-Rest + Logistic Regression** – demonstrated surprisingly strong performance, achieving a **Recall@5 of 0.87**, **Recall@7 of 0.93**, and a single-label accuracy of 87.43%. While slightly behind transformer models on key ranking metrics, these approaches were significantly faster to train and required substantially fewer computational resources. Their simplicity, transparency, and compatibility with CPU-only environments make them appealing for real-world deployment, especially in scenarios where interpretability, reproducibility, or infrastructure constraints are critical.
The strong performance of **Logistic Regression with TF-IDF** can be attributed to the alignment between the model's linear nature and the sparse, high-dimensional feature space generated by TF-IDF. In this representation, each term carries interpretable and informative weight, which Logistic Regression leverages to learn robust decision boundaries. The use of n-grams further enhanced performance by capturing short technical phrases (e.g., "timing error", "power loss") that are predictive of specific issue categories. This tight alignment between feature design and model capacity likely contributed to effective generalisation, even in the presence of incomplete label annotations.
Notably, the performance gap between the top classical pipeline and DistilBERT was moderate – approximately 5.8% lower on Recall@5 and 2.7% lower on Recall@7. While such differences may be critical in certain high-stakes applications, in this context they suggest that well-engineered lexical features can still enable classical models to uncover meaningful patterns in domain-specific technical texts. These results are consistent with recent findings by Veeranki et al. [59], who showed that traditional pipelines can remain competitive with transformers in structured, non-open-domain classification tasks.
Several factors likely contributed to the classical model's strong results in general. First, the structured and concise nature of the input (titles and descriptions) is well suited to sparse lexical representations. In such cases, the added complexity of deep contextual models may offer limited additional benefit. Second, the lower risk of overfitting, faster inference, and ease of maintenance make these models strong candidates for deployment in production environments where model efficiency and explainability are prioritised.
In summary, while transformer-based models offer superior semantic understanding and top-tier recall, classical pipelines such as TF-IDF combined with Logistic Regression remain practical and robust alternatives. They provide a valuable balance between performance, interpretability, and deployability, particularly in industrial settings where retraining speed, resource efficiency, and transparency are essential considerations.
In contrast, classical pipelines - particularly **TF-IDF (n-grams) + One-vs-Rest + Logistic Regression** – offered competitive performance with a **Recall@5 of 0.87** and single-label

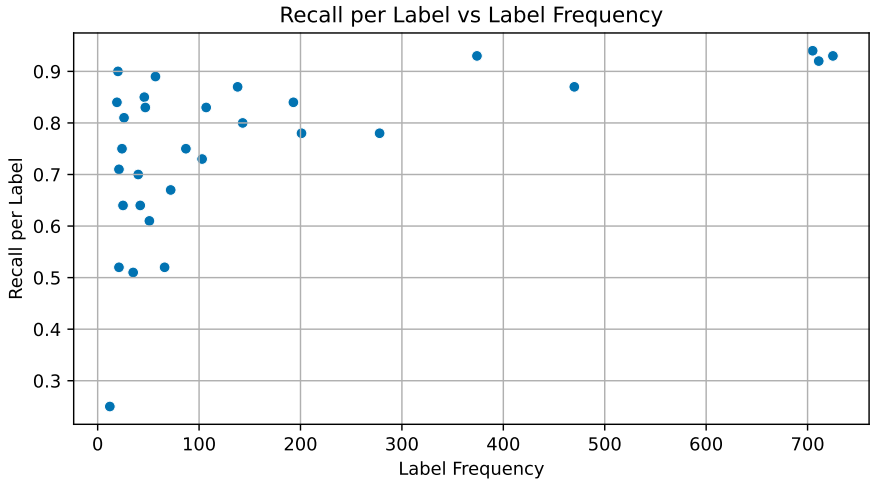


Figure 4.9: Per-label Recall plotted against label frequency TF-IDF vectorisation + One-vs-Rest + Logistic Regression. The plot shows a clear positive trend, with high-frequency labels achieving higher recall. However, several low-frequency labels still yield competitive recall, indicating that the model generalises reasonably well despite label imbalance.

accuracy of 87.43%. Their major advantage lies in training efficiency, interpretability, and deployability in CPU-only environments. These models remain suitable for use in resource-constrained or latency-sensitive settings. Interestingly, their performance was found to be comparable to that of transformer models. This observation aligns with recent work (Veeranki et al., 2024), where it was shown that classical models can remain competitive when supported by well-engineered feature extraction methods.
However, both categories of models revealed several systemic challenges.

- **Label Frequency Imbalance:** The scatter plots in Figure 4.9 and Figure 4.10 show a moderate positive correlation between label frequency and recall for both the best-performing classical model (TF-IDF + OvR + Logistic Regression) and the top transformer-based model (DistilBERT). As expected, frequent labels are generally recalled more reliably, while rare or long-tail labels are often underpredicted. However, several exceptions exist: a number of infrequent labels still achieve relatively high recall. This suggests that while label frequency is an important factor, it is not sufficient to determine recall performance on its own. Instead, factors such as label distinctiveness, contextual clarity, and semantic uniqueness of the associated text likely play a crucial role in improving retrieval performance for these rare but well-separated classes. Therefore, both frequency and semantic characteristics must be taken into account when interpreting or improving label-wise performance in these multi-label classification tasks.
- **Label Confusion and Semantic Overlap:** The confusion matrix for single-label predictions (Figure 4.11) reveals that frequent misclassifications occurred among certain

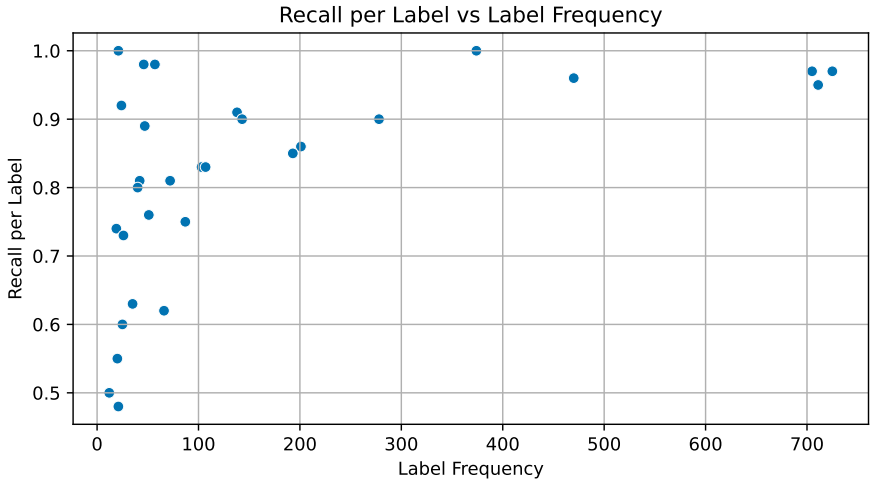


Figure 4.10: Per-label Recall plotted against label frequency for the DistilBERT model. The plot shows consistently high recall across both frequent and rare labels, with several low-frequency labels achieving recall above 0.8. This indicates DistilBERT's strong generalisation capacity in the presence of label imbalance.

categories, such as *software*, *architecture*, and *design*. These labels appear to be especially prone to confusion, likely due to overlapping terminology or similar contexts in the input text. Although this matrix is based on predictions from the best-performing classical pipeline (TF-IDF + OvR + Logistic Regression), a nearly identical pattern was observed with the DistilBERT model (see Appendix A). This suggests that these misclassification patterns are not specific to one architecture but rather stem from inherent ambiguity or semantic proximity between the labels themselves.

Figure 4.11 provides a heatmap visualisation of the model's predictions versus the true labels for single-label samples. Each row represents the true label, and each column shows the predicted label. The diagonal represents correct predictions, while off-diagonal cells indicate misclassifications. Darker cells reflect higher misclassification frequency. Notably, we observe recurring confusion between adjacent technical domains, such as *software* and *architecture*, or *design* and *ipdesign*, indicating that models occasionally struggle to separate conceptually close categories – especially in the absence of distinct language cues.

- **Incomplete Ground Truth and Metric Sensitivity:** Due to the known issue of incomplete multi-label annotations, precision and F1-scores were disproportionately low for all models, despite high recall. For example, DistilBERT achieved a macro F1 of only 0.3145, which does not fully reflect the model's practical effectiveness in retrieving relevant labels. As seen in Figure 4.12, while Recall@K increases steadily, Precision@K declines with larger K, emphasising the trade-off between comprehensive label recovery and prediction specificity.
- **Label Dependencies:** Figure 4.13 shows the label co-occurrence heatmap based

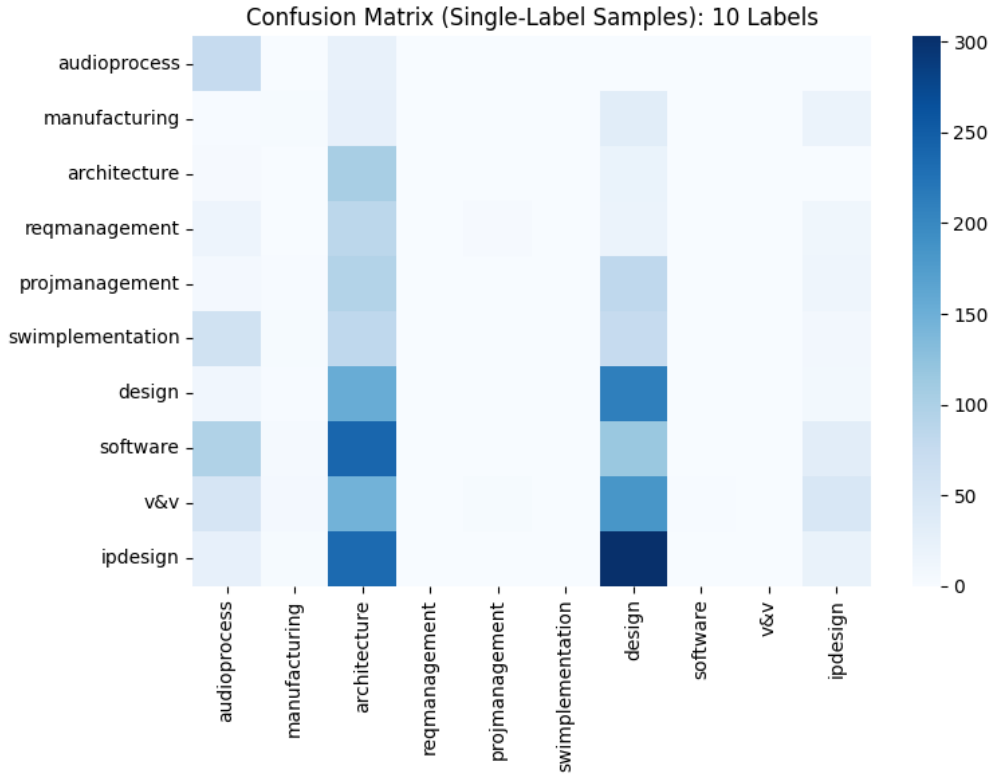


Figure 4.11: Confusion matrix for single-label samples (TF-IDF + Logistic Regression).

on multi-label samples from the training data. Each cell indicates how many times two labels appeared together in the same issue. While most label pairs co-occur infrequently, there are a few notable cases with higher co-occurrence counts – such as *dft&dxfx* with *radarprocess*, and *configmanagement* with *software* – which may reflect recurring relationships between certain engineering domains. These patterns suggest the presence of mild label dependencies that could be leveraged by models capable of capturing contextual or semantic co-labelling tendencies.

4.4.2 Explaining Unexpected Results

Several empirical outcomes diverged from initial expectations and merit deeper interpretation:

- **TF-IDF + Label Powerset + Logistic Regression** achieved the highest multi-label match score (78.52%) but the lowest single-label accuracy (85.12%). This discrepancy likely stems from Label Powerset's tendency to overfit to the specific label combinations seen during training. While it can precisely memorise known multi-label patterns, it struggles to generalise to new or unseen combinations – a limitation that is particularly problematic in evolving taxonomies or open-domain deployments.

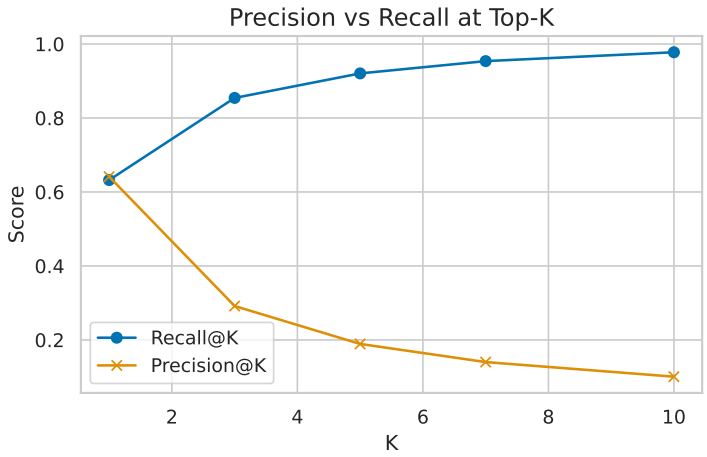


Figure 4.12: Precision and Recall at different top-K values (K = 1, 3, 5, 7, 10) for the DistilBERT model. As K increases, recall improves significantly, nearing 1.0 at K=10, while precision decreases due to incomplete ground truth. This trade-off highlights the importance of selecting an appropriate K based on application-specific tolerance for false positives versus missed relevant labels.

- **The addition of n-grams** generally improved ranking performance (e.g., MAP) but did not consistently lead to gains in overall accuracy. This was particularly evident in the **TF-IDF + OvR + Logistic Regression** pipelines, where the n-gram variant achieved the highest MAP score (0.6448) due to its ability to capture short, domain-specific phrases that are highly predictive of certain labels. However, this gain in ranking came at the cost of slightly reduced multi-label accuracy compared to the unigram-only variant. The n-gram model performed better on single-label examples, suggesting improved lexical precision in simpler cases. In contrast, the unigram model generalised better in multi-label scenarios, likely due to reduced sparsity and lower risk of overfitting. These results highlight a trade-off: while n-grams enhance discriminative power in phrase-level matching, they may introduce brittleness in sparse or noisy settings.

- **Naive Bayes + CountVectorizer** consistently outperformed its TF-IDF counterpart. This is somewhat unexpected, as TF-IDF is often more effective with linear models. However, Naive Bayes assumes feature independence and relies on raw term counts rather than relative weighting – assumptions that align better with CountVectorizer's output. This pairing may thus preserve term frequency signals that are more informative for NB than TF-IDF's normalised scores.

- **Multinomial Naive Bayes + One-vs-Rest** yielded the strongest results among NB configurations, which confirms the benefit of decomposing multi-label classification into binary subproblems when using probabilistic classifiers. The OvR strategy al-

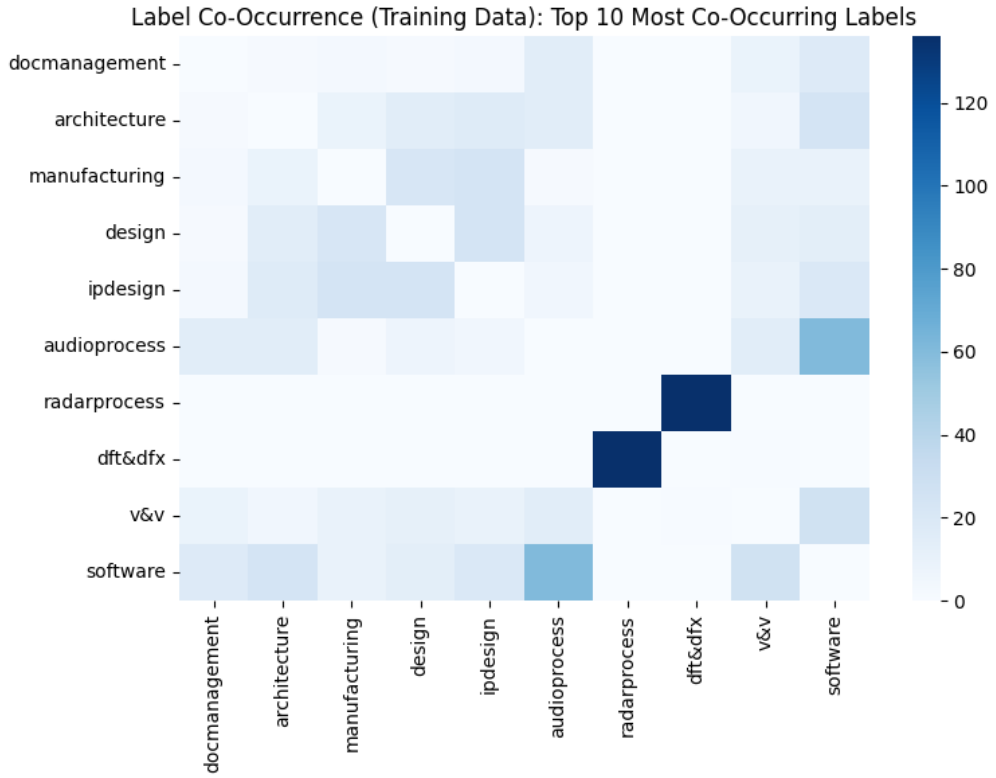


Figure 4.13: Label Co-Occurrence Heatmap (Training Data).

lows Naive Bayes to focus on isolated label-specific distributions, avoiding combinatorial complexity and label interaction effects that Naive Bayes struggles to handle effectively.

- **Minimal performance gains from more complex classical models:** Models such as Support Vector Machines (LinearSVC), Ridge Classifier, Random Forest, XGBoost, and k-Nearest Neighbours performed noticeably worse than simpler baselines like Logistic Regression and Naive Bayes. This is somewhat unexpected and suggests that the engineered features (TF-IDF, CountVectorizer) are already linearly separable, rendering additional model complexity unnecessary or even detrimental.

- **Recall@1 vs. Recall@10:** Classical models showed significantly lower Recall@1 compared to transformers, suggesting they are less precise in ranking the single most relevant label. However, Recall@10 scores across many models were similar, indicating that classical methods can still retrieve the most relevant labels but rank them less effectively. This highlights the value of transformers in tasks that require high precision at the top of the list, such as decision support, while classical models may suffice for exploratory or bulk tagging.

- **Deep learning (ANN, MLP, CNN, RNN)** models performed worse than expected despite being paired with dense embeddings like Word2Vec and FastText. This may be due to a mismatch between model complexity and dataset size. These architectures typically require large, diverse datasets and extensive tuning conditions not met in this use case. Moreover, without significant sequence-level dependencies in the data (i.e., long contextual flow), these models may have failed to exploit their full capacity.
- **Embedding-based features underperformed sparse representations:** Word2Vec, FastText, GloVe, and SBERT embeddings generally produced weaker results than TF-IDF or CountVectorizer. This could be due to domain mismatch (pretrained embeddings not optimised for technical language) or the lack of sequence-level supervision needed to fully exploit these dense vectors. Additionally, sparse representations preserve explicit lexical signals, which appear to be more informative in this structured, domain-specific task.

In summary, while transformer models delivered state-of-the-art results, several unexpected findings reinforce the enduring relevance of simple yet well-matched classical approaches. Careful attention to problem transformation strategies, feature-model compatibility, and data characteristics proved more important than model complexity in many cases—offering key insights for practical deployment in industrial settings.

4.4.3 Model Trade-offs and Deployment Implications

While **DistilBERT** is clearly the best-performing model across ranking and multi-label accuracy metrics, it comes with higher computational cost and increased deployment complexity. Fine-tuning and inference with transformer-based models require GPU resources for acceptable latency, and memory consumption can be substantial, making real-time or large-scale deployment more challenging without dedicated infrastructure. Furthermore, transformer predictions are often less interpretable, which may hinder adoption in safety-critical or regulated engineering environments where traceability and model reasoning are important.

In contrast, classical models such as **TF-IDF + Logistic Regression** offer a favourable balance between recall and coverage while maintaining much lower computational demands. These models can be trained and deployed entirely on CPU, making them suitable for integration into existing systems without requiring hardware upgrades. Their transparency also allows for easier explanation of predictions based on term-level contributions, which can support auditing, debugging, or user-facing justification of results.

Additionally, the choice of classification strategy impacts practical trade-offs. **One-vs-Rest (OvR)** classifiers treat each label independently, enabling modular training and easier scalability to new or updated labels, but they may fail to capture relationships between co-occurring categories. In contrast, **Label Powerset (LP)** approaches treat each unique label combination as a single class, enabling better modelling of co-labelling patterns. In this study, LP achieved the highest exact multi-label match score (78.52%) among classical configurations. However, LP models are less flexible: they cannot generalise to unseen label combinations during inference and require retraining when new combinations emerge. This

rigidity, combined with higher training complexity, makes them more difficult to maintain in dynamic production environments.
For industrial use cases, the choice of model should therefore consider not only predictive performance but also inference time, infrastructure constraints, ease of integration, and maintainability. In settings where real-time predictions, explainability, and low-cost deployment are priorities, classical models remain a strong and practical option despite their slightly lower top-K recall scores.

4.4.4 Practical Implications and Deployment Relevance

The findings carry several important implications for deployment in industrial systems:

- **Trade-offs matter:** Industrial model selection must balance accuracy with integration cost, retraining overhead, explainability, and infrastructure compatibility – especially in dynamic, resource-sensitive engineering environments.
- **DistilBERT** is the strongest candidate when predictive performance is critical. Its high Recall@K and robust ranking ability make it ideal for batch annotation, issue triaging, or downstream recommendation tasks.
- **TF-IDF + Logistic Regression** offers a practical, interpretable, and low-cost solution. It is especially suitable for real-time systems, CPU-only environments, or embedded tools where explainability and latency are crucial.
- **Naive Bayes with CountVectorizer** delivers respectable performance with virtually no inference time, making it a strong baseline for CPU-constrained deployments or as part of an ensemble.

4.4.5 Conclusion and Answer to the Research Question

RQ: How can multi-label classification techniques be effectively applied to categorise engineering issue descriptions in a real-world industrial setting?
This study demonstrates that multi-label classification can be successfully applied to engineering issue tracking, provided that model selection is guided by both performance metrics and industrial constraints. Transformer-based models, particularly fine-tuned versions such as DistilBERT, showed strong results in terms of semantic relevance and top-K recall. These models are well-suited for handling rich and complex textual data, especially when high predictive accuracy is required.
At the same time, classical models – such as TF-IDF combined with Logistic Regression – proved to be competitive across multiple evaluation metrics. They offer advantages in terms of computational efficiency and interpretability, which are particularly important in resource-constrained or transparency-critical environments.
Overall, effective deployment of multi-label classifiers in industrial settings requires a careful balance between model complexity, accuracy, and operational feasibility. Evaluation frameworks must also account for domain-specific challenges such as label sparsity, co-occurring categories, and semantic ambiguity. By aligning model capabilities with real-world requirements, this work provides practical guidance for scalable, interpretable, and effective categorisation of technical issues in engineering workflows.

Chapter 5

Association

Engineering issue tracking systems hold valuable historical knowledge that, when effectively leveraged, can reveal recurring patterns, support faster resolution, and enhance knowledge reuse. One way to achieve this is by automatically discovering associations between semantically related issues. This chapter addresses the second core task of the research: identifying meaningful associations using NLP techniques. It defines the problem, outlines the methodology, and presents the implemented models. Two complementary evaluation strategies are described, and the results are analysed to compare model performance, assess expert and LLM agreement, and discuss practical implications for industrial deployment.

5.1 Problem definition

In engineering issue tracking systems, different reports often share similarities-whether in root causes, affected components, or resolution strategies. However, these relationships are rarely identified systematically, resulting in redundant reports, duplicated troubleshooting efforts, and inefficient resolution workflows. Engineers frequently encounter recurring issues across projects, but without automated tools for discovering such connections, they must rely on manual searches and personal experience.
The key challenges in discovering associations between issues include:

- **Lack of structured linking mechanisms**: Issue tracking platforms often store each ticket independently, without consistent cross-referencing to semantically related or historically similar issues.
- **High volume and diversity of reports**: Large engineering databases contain thousands of reports with varying levels of detail and structure, making manual association infeasible.
- **Semantic complexity**: Teams may use different terminologies or phrasing to describe similar issues, reducing the effectiveness of simple keyword-based matching.
- **Scalability concerns**: Traditional search and retrieval methods degrade in performance as the dataset grows, necessitating more intelligent, scalable approaches.
- **Subjective relevance criteria**: The definition of what constitutes a "similar" issue varies between teams and tasks, complicating both model design and evaluation.

In an industrial context, these challenges are particularly acute. As engineering workflows grow more complex and interdisciplinary, the lack of reliable association discovery hinders knowledge sharing and slows down the resolution. Effective association discovery can reduce duplicated engineering efforts, support faster resolution of recurring problems, and enhance the reusability of insights embedded in historical issue data.
Currently, issue associations are added manually by engineers, but these links often reflect procedural steps (e.g., follow-up actions, demo preparation) rather than semantically related content. Consequently, the association field in the issue tracking system remains underutilised and does not support systematic retrieval of similar cases. Without automated support, engineers must rely on memory or manual keyword search, both of which are inefficient at scale.
To address these limitations, Natural Language Processing (NLP) offer a promising direction by enabling semantic comparison of issue descriptions. By identifying latent similarities between new and historical reports, NLP-based models can recommend relevant past tickets, uncover hidden relationships, and assist engineers in reusing knowledge more effectively.
This research focuses on designing and evaluating models for automated association discovery in issue tracking systems. Specifically, the models aim to:

1. Automatically suggest previously reported issues that are relevant to new tickets.
2. Support engineering workflows by providing quick access to historical resolutions, thereby reducing duplicated reports and effort.

This section addresses the second research question introduced in Chapter 1, which investigates how semantic associations between issue reports can be identified using NLP techniques and how the most relevant related issues can be retrieved for a given report.

5.2 Methodology

This section outlines the methodology developed to address the problem of automated discovery of semantically related engineering issues in large-scale industrial datasets. The approach integrates data collection, preprocessing, validation of association labels, and implementation of a diverse set of similarity-based and clustering-based methods. Given the absence of complete association annotations and the subjective nature of similarity, the evaluation framework combines both automatic metrics and qualitative assessment by domain experts and an LLM-based assistant. The primary goal is to maximise the retrieval of relevant associations while ensuring that the results are useful in practice and consistent with expert evaluations.

5.2.1 Dataset

The initial dataset consists of 63,329 engineering issue reports. Each record includes an `Artifact ID`, a short `Title`, a detailed `Description`, and optionally a list of manually curated `Associations`. Only these four fields were retained for the association task. This

choice reflects the real-world constraints of the intended use case, where automatic recommendations must rely solely on textual information typically available at the time of issue creation. No structured metadata or resolution history was used to ensure the approach remains broadly applicable and practically deployable.

A data cleaning pipeline consistent with the classification task (Section 4.2.2) was applied. Of the 63,329 issues, 7,735 contained at least one manually assigned association. Each association was validated to ensure that it referenced another issue within the dataset. Associations pointing to external or missing records were discarded. This resulted in a final set of 6,378 valid associations.

5.2.2 Implemented Techniques

Different techniques were implemented to explore diverse paradigms for identifying semantically related issues. These methods were grouped into three broad categories: lexical retrieval, embedding-based similarity, and clustering-based filtering. Each method produced a ranked list of candidate associations for a given issue, based on textual similarity, shared topic membership, or spatial proximity in vector space.

An overview of the data preparation and modelling stages, including the validation of association labels and retrieval of top-K predictions using various techniques, is illustrated in Figure 5.1.

Lexical retrieval methods

Lexical retrieval methods rely on exact or weighted word overlaps between issue texts. Three approaches were used in this category. The first, BM25, is a probabilistic ranking function that scores candidate issues based on the frequency and rarity of terms in the query and the candidate document. The second, TF-IDF + cosine similarity, constructs sparse vector representations of each issue using term frequency-inverse document frequency and compares them via cosine similarity. These methods prioritise surface-level lexical matches and are often effective for capturing terminology-based similarity. The third, CountVectorizer + cosine similarity, uses raw token frequency counts instead of TF-IDF weights for vector construction.

For both TF-IDF and CountVectorizer, n-gram ranges (unigrams and bigrams) were included to capture short multi-word expressions commonly used in engineering contexts. These methods prioritise surface-level lexical matches and are particularly effective for detecting terminology-based similarity, especially when technical phrasing is consistent across issues.

Embedding-based similarity methods

Embedding-based similarity methods aim to capture semantic relationships by projecting issue texts into dense vector spaces where semantically similar texts are close together. Four word embedding techniques were explored: SBERT [47], Word2Vec [39], FastText [6], and GloVe [43].

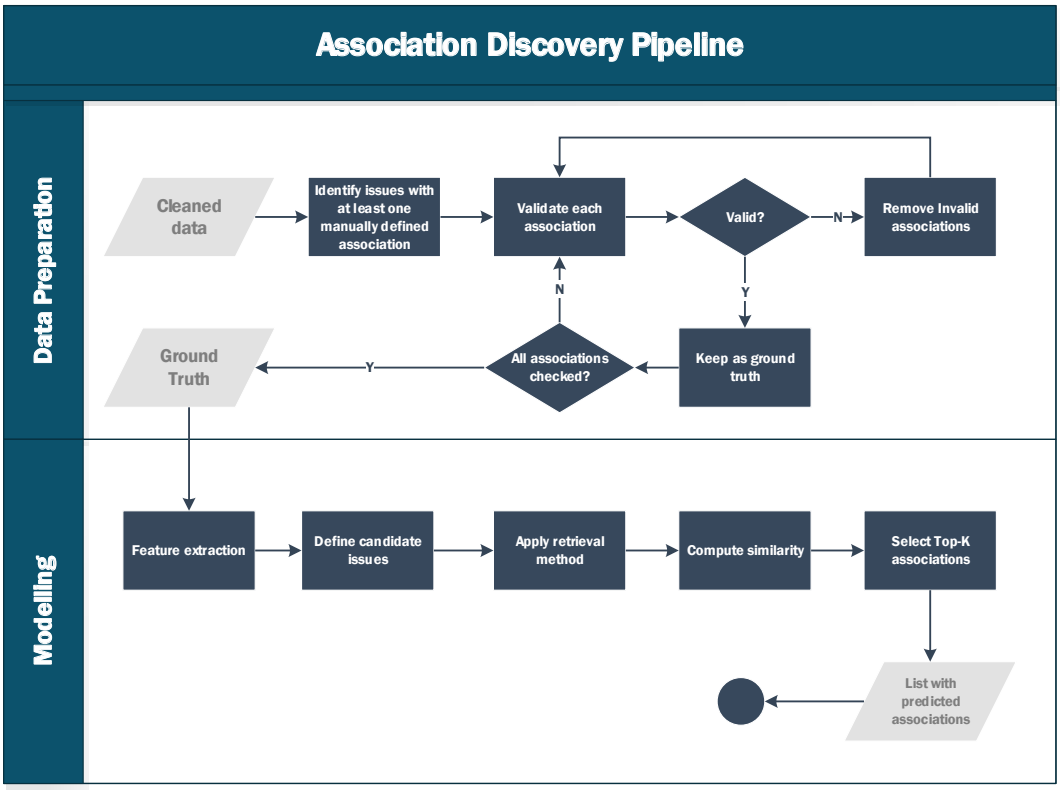


Figure 5.1: Overview of the association discovery pipeline. The process consists of two stages: (1) **Data Preparation**, where manually defined associations are validated and filtered to form a ground truth set, and (2) **Modelling**, where candidate associations are generated using similarity-based retrieval methods and the top-K most similar items are selected as predictions.

SBERT + cosine similarity uses the Sentence-BERT model to encode each issue (title + description) into a fixed-size sentence embedding. These embeddings capture contextual and syntactic information and are compared using cosine similarity.

In contrast, Word2Vec, FastText, and GloVe rely on static word embeddings and provide a more lightweight alternative. These methods are less sensitive to word order but remain effective at capturing core semantic content. Cosine similarity was used to compare the resulting vectors and rank candidate issues accordingly.

These embedding-based methods are generally more robust than lexical approaches to synonymy, morphological variation, and paraphrasing, making them well-suited for noisy or inconsistently worded engineering data.

Clustering-based filtering methods

Clustering-based filtering methods aim to narrow down the search space by first grouping issues based on textual similarity and then ranking candidates within the same cluster.

Multiple clustering techniques were employed. TF-IDF representations were clustered using DBSCAN and k-Means, with cosine similarity applied within clusters to select the top candidates. A similar strategy was followed for Word2Vec + k-Means, which used dense embeddings for clustering. In addition, two topic modelling approaches were explored: Latent Dirichlet Allocation (LDA), which assigns issues to latent topics based on word distributions, and BERTopic, which applies topic modelling on SBERT embeddings. These methods are designed to filter recommendations to semantically coherent subgroups before applying finer-grained similarity scoring.
All methods returned a fixed number of top-k candidate issues per query (with k typically set to 10 during evaluation). The goal was to assess how effectively each technique could recover known associations among a large corpus of issues using only their title and description fields.

5.2.3 Evaluation

To assess the effectiveness of the implemented methods for association discovery, a two-phase evaluation strategy was employed: (1) a **quantitative evaluation** against ground-truth links, and (2) a **qualitative evaluation** based on domain expert assessments and LLM-based scoring of semantic relevance.
The evaluation strategy and subsequent recommendation workflow are summarised in Figure 5.2. It illustrates the two-phase evaluation framework, quantitative and qualitative, as well as the decision-making steps leading to model selection.

Quantitative Evaluation

The quantitative evaluation assessed each method's ability to retrieve known associations from a labelled set of 6,378 validated issue links. For every issue, the top-10 predicted associations were generated and compared to the ground-truth link(s). The following metric was used to measure performance:

Hit@k
Hit@k measures the proportion of issues for which the ground-truth associated issue appears in the top-k predicted results.

$$\text{Hit@}k = \frac{1}{N}\sum_{i=1}^{N}\text{hit}_i \tag{5.1}$$

where $\text{hit}_i = 1$ if the ground-truth association for issue i is among the top-k predictions, and 0 otherwise.
Hit@k is a widely used and interpretable metric in ranking-based evaluations, particularly suited for scenarios like issue recommendation, where the task is to retrieve relevant associations from a large candidate pool. In this context, each issue has exactly one associated issue, and the goal is to determine whether this correct association appears among the top-k suggestions. This makes Hit@k especially appropriate, as it directly reflects the system's ability to surface the correct result. A higher Hit@k score indicates that the method more

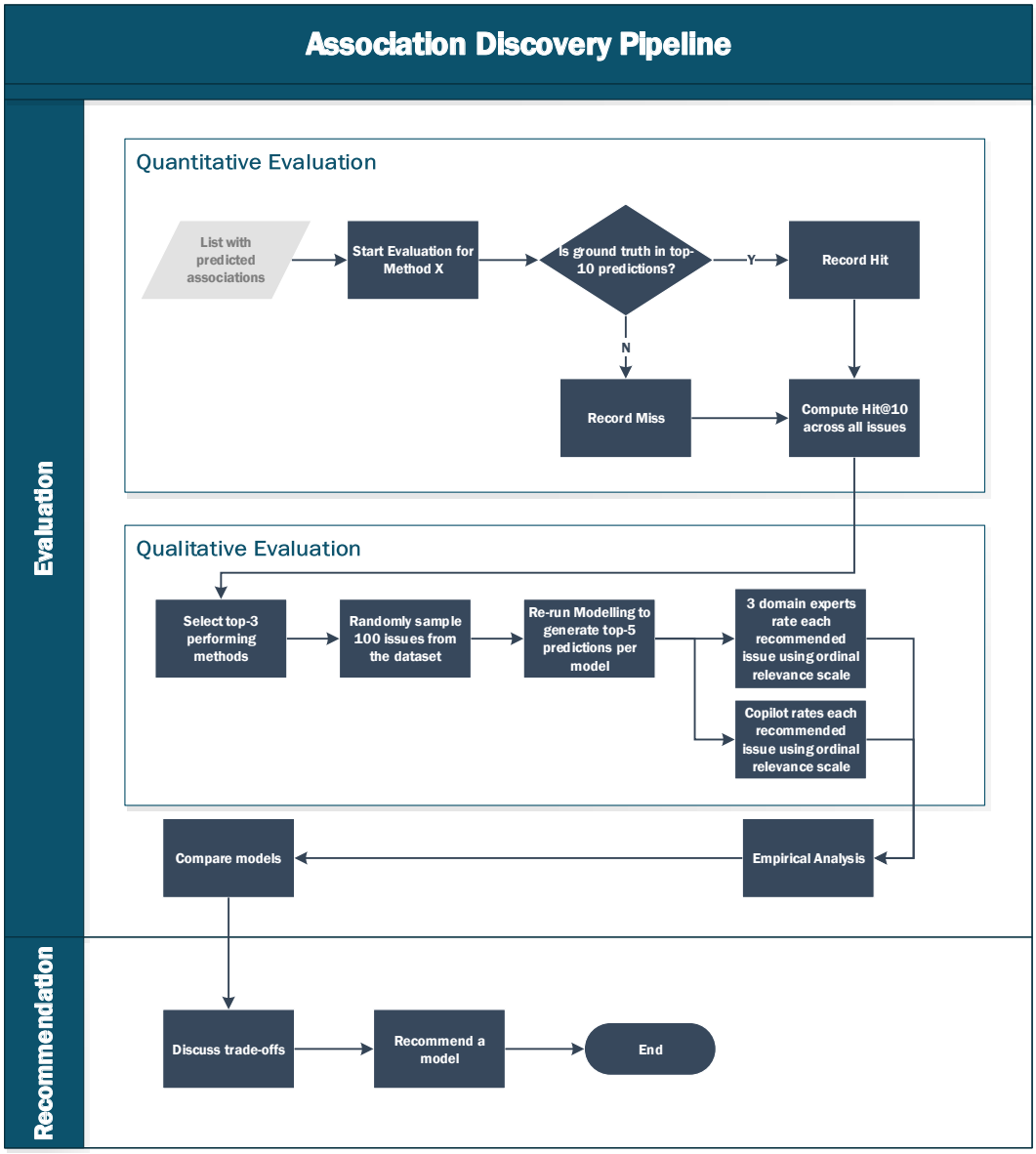


Figure 5.2: Extension of the association discovery pipeline focused on **Evaluation** and **Recommendation**. The top section shows the **quantitative evaluation**, which computes Hit@10 by checking whether ground truth associations are ranked among the top-10 predictions. The middle section outlines the **qualitative evaluation**, where the top-3 performing methods are re-evaluated by three domain experts and an AI assistant (Copilot). The final section presents the **recommendation process**, comparing model performances and discussing trade-offs to select a preferred method for deployment.

frequently ranks the correct association within the top-k recommendations. In this study, $k = 10$ was selected to reflect realistic retrieval scenarios where engineers are presented with a shortlist of similar issues.

Sample Selection for Qualitative Evaluation

To conduct a deeper semantic evaluation, a statistically representative sample of 100 issues was drawn from the labelled dataset using simple random sampling. The required sample size was computed with a 95% confidence level and a 10% margin of error using the standard formula for estimating proportions [25]:

$$n = \frac{Z^2 \cdot p \cdot (1-p)}{e^2} = \frac{1.96^2 \cdot 0.5 \cdot (1-0.5)}{0.1^2} = \frac{3.8416 \cdot 0.25}{0.01} = 96.04 \tag{5.2}$$

where:

- Z is the z-score for a 95% confidence level (1.96),
- p is the estimated proportion of relevant results (conservatively set to 0.5 to assume maximum variability),
- e is the margin of error (set to 0.1).

Since the population is finite ($N = 6,378$), a finite population correction (FPC) was applied:

$$n_f = \frac{n}{1 + \frac{n-1}{N}} = \frac{96.04}{1 + \frac{95.04}{6378}} \approx \frac{96.04}{1.0149} \approx 94.58 \tag{5.3}$$

To ensure robustness and account for possible anomalies, the sample size was rounded up to 100 issues.

Qualitative Evaluation

To complement the automated metrics, qualitative assessments were conducted on the top-3 performing models. Each model produced top-5 ranked recommendations for the 100 sampled issues. These were then evaluated by three human experts and an LLM-based system.

Expert-Based Relevance Rating

Three domain experts independently rated each recommended issue on a 4-point ordinal scale:

- **2 – Useful:** Clear semantic or technical similarity.
- **1 – Partially Useful:** Some shared context or components.
- **0 – Not Useful:** No semantic or technical connection.
- **-1 – Not Enough Information:** The issue description lacks sufficient detail to make a relevance judgment.

The average relevance score was computed for each method to facilitate direct comparison of perceived semantic quality. To further analyse the consistency of expert assessments, inter-rater agreement was evaluated using established reliability metrics.

LLM-Based Evaluation

A large language model (Copilot[1]) was used to rate the same set of top-5 recommendations for the top-3 methods as the human experts. The model was accessed via its online chatbot interface and instructed through structured prompts to assess semantic relevance on the same 4-point scale.

The prompt described the task and the rating categories as follows: *"You are given a source issue (title and description) and a list of 15 recommended issues (each with a title and description). Your task is to assess how relevant each recommended issue is to the source issue. The goal is to support engineers in identifying past issues whose solutions may help resolve the current one. For each recommendation, choose one of the following labels:* ***Useful*** *– clearly relevant and likely to help resolve the current issue;* ***Maybe useful*** *– possibly relevant but uncertain or only partially helpful;* ***Not useful*** *– unrelated or unlikely to help;* ***I don't know*** *– not enough context or unfamiliar topic to decide."*

This evaluation offered an additional, automated perspective on semantic similarity and tested the feasibility of using LLMs as scalable evaluators in an industrial context. Agreement between Copilot and the expert engineers was measured by comparing rating distributions, average relevance scores, and overlaps in top-ranked recommendations.

Agreement between Copilot and the expert engineers was measured by comparing rating distributions, average relevance scores, and overlaps in top-ranked recommendations.

Inter-Rater Agreement: Krippendorff's Alpha

Krippendorff's Alpha (α) is a statistical measure of inter-rater reliability that accounts for chance agreement and supports multiple raters, different data types (nominal, ordinal, interval), and missing values. It is particularly well-suited for ordinal-scale relevance judgments, as in this study. The metric is defined as:

$$\alpha = 1 - \frac{D_o}{D_e} \tag{5.4}$$

where D_o is the observed disagreement among raters, and D_e is the expected disagreement by chance. An α value of 1 indicates perfect agreement, while 0 indicates agreement no better than random chance. In practice, the interpretation scale for Krippendorff's Alpha is as follows: $\alpha \geq 0.80$ indicates strong agreement; $0.67 \leq \alpha < 0.80$ suggests substantial or tentative agreement; $0.40 \leq \alpha < 0.67$ indicates moderate agreement; $0.20 \leq \alpha < 0.40$ reflects slight agreement; and $\alpha < 0.20$ is considered poor or no agreement. These thresholds help contextualise the level of consistency among raters and inform the reliability of subjective evaluations [26].

Statistical Comparison and Correlation Analysis

While these average scores already indicate preference variations across models and evaluators, it is important to assess whether these differences are statistically meaningful. To assess statistical differences between the relevance ratings of different methods, the **Kruskal-**

[1] https://copilot.microsoft.com/ The evaluation was conducted in June 2025, and due to the evolving nature of LLMs, future versions of Copilot may produce different results

Wallis H test was applied [27]. This non-parametric test evaluates whether the distributions of scores differ significantly across multiple groups, without assuming normality.
To explore correlations between human ratings, LLM scores, and model rankings, two correlation coefficients were used: **Pearson's** r and **Spearman's** ρ. Pearson's r measures linear correlation between continuous variables [11], while Spearman's ρ captures monotonic relationships by ranking the data [52]. Both coefficients range from -1 (perfect negative correlation) to +1 (perfect positive correlation), with values near 0 indicating no correlation.
Additionally, **Jaccard similarity** was used to quantify the degree of overlap between the sets of associations retrieved by different models. It is defined as the size of the intersection divided by the size of the union of two sets [22], providing a measure of how similar two models' outputs are in terms of shared recommendations.

5.3 Results

This section presents the results of both the automatic and human-in-the-loop evaluations used to assess the effectiveness of the proposed association discovery methods. We begin with a quantitative analysis based on the retrieval performance over known associations, followed by a detailed qualitative evaluation based on expert ratings and comparisons with a large language model. The results are further examined in terms of inter-rater agreement, rank relevance, model output diversity, and prediction overlap to provide a comprehensive understanding of model behaviour and retrieval quality.

5.3.1 Automatic Evaluation

The performance of each method was quantitatively evaluated based on its ability to recover known associations from the labelled dataset. Specifically, for each issue with a valid associated issue, it was verified whether the correct association appeared within the top-10 most similar results produced by the method. Table 5.1 summarises the number and percentage of successful retrievals for each method.
The best-performing method was the classical information retrieval model **BM25**, which successfully retrieved 3,277 out of 6,378 known associations, achieving a Hit@10 score of 51.4%. Close behind were two semantically-informed methods: **SBERT + Cosine Similarity** (48.1%) and **SBERT + BERTopic** (47.6%). These models leverage transformer-based embeddings to capture contextual meaning in issue descriptions, enabling more accurate identification of semantically related reports.
The clustering-based approach **TF-IDF + DBSCAN + Cosine Similarity** also demonstrated competitive performance, retrieving 3,004 associations (47.1%). Several other embedding- and clustering-based methods yielded moderate results, with accuracy ranging between 37% and 43%.
At the lower end of the performance spectrum, simpler methods such as **Word2Vec + Cosine Similarity** (36.0%) and **TF-IDF + Cosine Similarity** (37.0%) underperformed, indicating that shallow lexical similarity models are less effective in capturing the nuanced relationships present in engineering issue descriptions.

Method	Associations Found (Top-10)
BM25	**3277 / 6378 (51.4%)**
SBERT + Cosine Similarity	3069 / 6378 (48.1%)
SBERT + BERTopic	3035 / 6378 (47.6%)
TF-IDF + DBSCAN + Cosine Similarity	3004 / 6378 (47.1%)
TF-IDF + k-Means + Cosine Similarity	2707 / 6378 (42.4%)
FastText + Cosine Similarity	2588 / 6378 (40.6%)
FastText + DBSCAN + Cosine Similarity	2582 / 6378 (40.5%)
GloVe + Cosine Similarity	2514 / 6378 (39.4%)
LDA	2472 / 6378 (38.8%)
TF-IDF (n-grams) + Cosine Similarity	2454 / 6378 (38.5%)
Word2Vec + k-Means + Cosine Similarity	2403 / 6378 (37.7%)
TF-IDF + Cosine Similarity	2360 / 6378 (37.0%)
Count Vectorizer (n-grams) + Cosine Similarity	2347 / 6378 (36.8%)
Count Vectorizer + Cosine Similarity	2283 / 6378 (35.8%)
Word2Vec + Cosine Similarity	2294 / 6378 (36.0%)

Table 5.1: Top-10 association retrieval performance across different methods. The table reports the number and percentage of correct associations retrieved within the top-10 predictions out of 6378 total ground-truth associations.

Overall, these results highlight the effectiveness of embedding-based and retrieval-driven methods over traditional lexical similarity approaches. However, even the strongest model retrieved just over half of the known associations, underscoring the difficulty of the task and motivating the need for further evaluation.

A post-hoc analysis revealed that **2,077 of the 6,378 valid associations were not retrieved by any method**, suggesting that a substantial portion of ground-truth links remain challenging to detect using current methods. Upon closer inspection of a sample of these cases – manually reviewed by an engineer – many of the unretrieved links were identified as *procedural associations*. These are associations that arise not from shared terminology or content similarity, but from procedural dependencies in the engineering process, such as issues being linked because they belong to the same test flow, product release, or design milestone. Such relationships are often implicit and context-driven, and may not be explicitly reflected in the textual descriptions. This finding highlights a key limitation of content-based retrieval methods and underscores the need for complementary evaluation strategies that integrate human judgment and context-aware interpretation, as further explored in the qualitative evaluation.

5.3.2 Qualitative Evaluation

To complement the automatic evaluation, a qualitative review was conducted to assess the semantic quality and practical relevance of the top-ranked associations. This phase involved

domain experts[2] as well as a large language model (Copilot), offering a human-in-the-loop perspective on model output quality. High textual similarity scores do not necessarily translate into usefulness from an engineering perspective – many top-ranked associations, while lexically or semantically similar, may be irrelevant in practice. This reinforces the importance of human evaluation and context-sensitive analysis in assessing the real-world applicability of retrieval methods. The qualitative analysis focused on the three best-performing methods from the quantitative phase – **BM25**, **SBERT + Cosine Similarity**, and **SBERT + BERTopic**. These models not only achieved the highest scores but also represent a diverse range of retrieval paradigms: lexical, embedding-based, and topic-aware semantic approaches.

Annotation Coverage and Sample Size

Three engineers participated in the manual evaluation, but annotation coverage varied across individuals. Engineer 1 and Engineer 3 each reviewed the complete set of **1,500 suggestions** (500 per model), whereas Engineer 2 assessed a smaller subset of **790 associations** – comprising 280 suggestions from BM25, and 255 each from Cosine Similarity and BERTopic. This uneven distribution introduces a potential source of bias, particularly when comparing rating distributions, computing inter-rater agreement, or averaging scores across raters. As such, results involving Engineer 2 should be interpreted with caution, as the reduced sample size may affect the reliability of the results.

Rating Distributions Across All Evaluators

The overall distribution of ratings from all human evaluators and Copilot is shown in Figure 5.3. When including Copilot, 39% of recommendations were rated as *not useful*, 23% as *partially useful*, and 35% as *useful*. The remaining 3% received a score of *not enough information*, indicating uncertainty.
When Copilot scores are excluded, the proportion of *not useful* ratings increases to 45%, while *partially useful* drops to 17%, with the *useful* category remaining relatively stable. The low frequency of *not enough information* responses ($< 4\%$) across both human and LLM evaluations suggests that the vast majority of recommended associations were interpretable and assessable by both humans and the language model.
Figure 5.4 illustrates the normalised rating distributions across the three engineers and the Copilot. Engineer 1 consistently provided more favourable evaluations, rating 46% of suggestions as *useful* and assigning only 2% as *not enough information*. In contrast, Engineer 2 exhibited a broader spread, with 28% marked as *useful*, 14% as *not enough information*, indicating a more cautious or uncertain evaluation pattern. Engineer 3 applied the most strict criteria, assigning the highest proportion of *not useful* ratings (56%), along with 18% for *maybe useful* and only 0.3% *not enough information*, suggesting a more decisive but critical evaluation style.

[2]This evaluation involved a small number of professional engineers from the industry who voluntarily assessed the output of automated models in the context of their domain expertise. No personal or sensitive data was collected, and participants were not subjected to any form of intervention or manipulation. As such, formal ethics approval was not required in accordance with TU Delft's research ethics policy for low-risk studies.

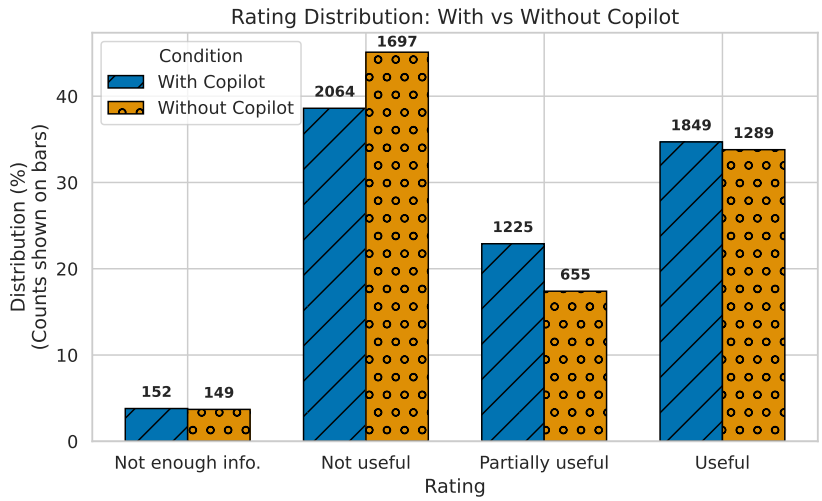


Figure 5.3: Distribution of usefulness ratings for association suggestions, shown separately for conditions including and excluding Copilot. The bars display percentage distributions with raw counts annotated.

Copilot, by comparison, demonstrated the most balanced rating distribution: 38% of suggestions were rated as *maybe useful*, 37% as *useful*, 24% as *not useful*, and only 0.2% as *not enough information*. These trends reveal substantial inter-rater variability, which plays a key role in interpreting model performance and evaluating consistency across human and automated assessments.

Rating Distributions by Model

Figure 5.5 and Figure 5.9 provide a comparative overview of the rating behaviour across the three evaluated models, as judged by the engineers and Copilot.

Among the human evaluators (Figure 5.5), BM25 received the highest proportion of *useful* labels (36%), followed by SBERT (34%) and BERTopic (32%). Conversely, BERTopic had the highest share of *not useful* ratings (48%), suggesting that its recommendations were more frequently judged irrelevant. The share of *not enough information* responses remained low across all models (below 4%), indicating that engineers were generally confident in making judgments.

Copilot's ratings, shown in Figure 5.9, followed a different pattern. While its *useful* ratings remained high (between 34.8% and 40.2%), the *maybe useful* label was its most common rating, exceeding 39% for BM25 and BERTopic. Unlike human evaluators, Copilot issued virtually no *not enough information* responses, reflecting a tendency to always assign interpretive value, even in ambiguous cases.

Taken together, these patterns suggest that BM25 was slightly better received by human evaluators, while Copilot exhibited a preference for more moderate judgments.

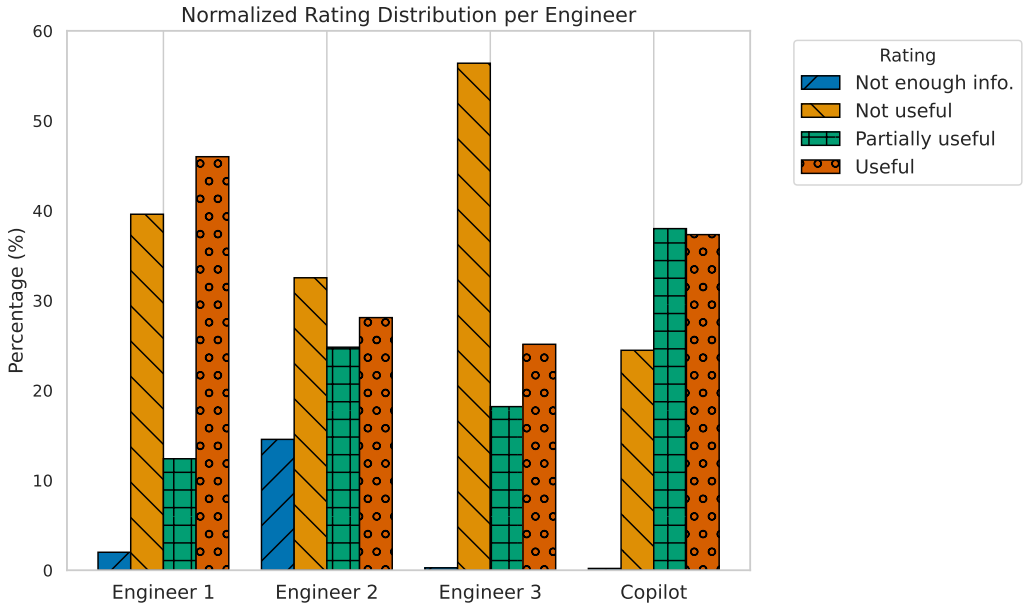


Figure 5.4: Normalised distribution of usefulness ratings per Engineer and Copilot. The plot shows percentage breakdowns for each rating category: Not enough information, Not useful, Partially useful, and Useful.

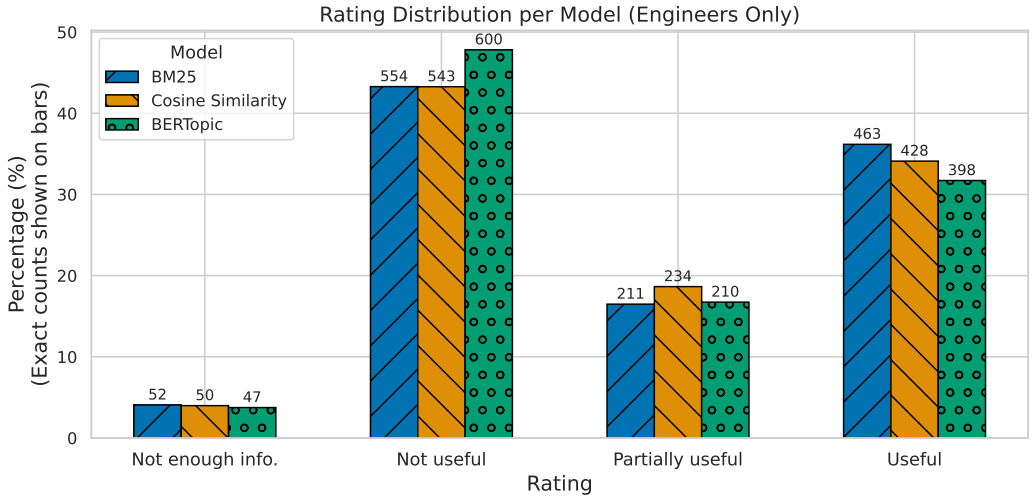


Figure 5.5: Distribution of usefulness ratings provided by engineers for the three models (BM25, Cosine Similarity, BERTopic). Ratings are categorised as Not enough information, Not useful, Partially useful, and Useful.

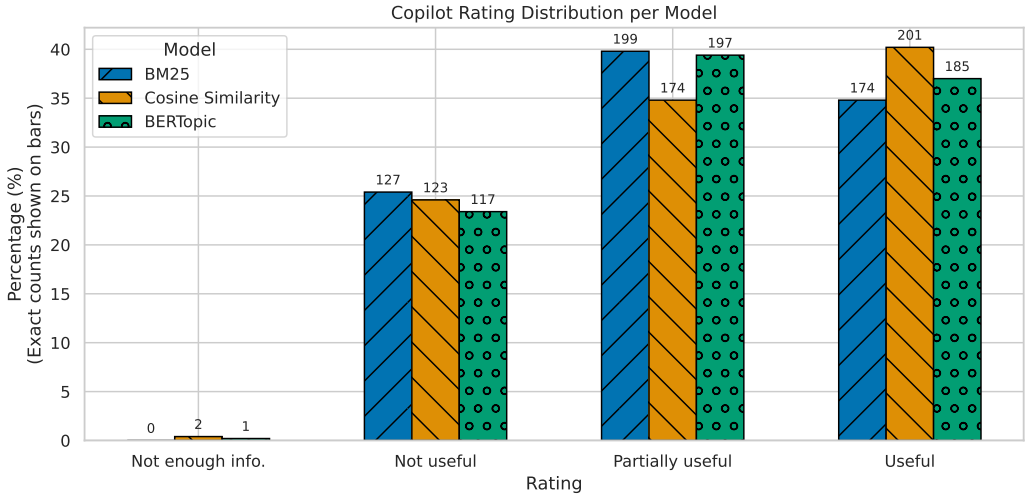


Figure 5.6: Distribution of usefulness ratings provided by Copilot for the three models. Ratings are grouped as Not enough information, Not useful, Partially useful, and Useful.

Inter-Rater Agreement and Correlation Analysis

Inter-rater reliability was assessed using Krippendorff's Alpha (α). Agreement among the three engineers yielded $\alpha = 0.53$, which corresponds to **moderate** agreement[3]. When Copilot ratings were included, the overall agreement decreased slightly to $\alpha = 0.51$.

When broken down by model, BERTopic achieved the highest agreement among human raters ($\alpha = 0.58$), followed by Cosine Similarity ($\alpha = 0.54$) and BM25 ($\alpha = 0.47$). Interestingly, adding Copilot ratings increased α for BM25, while slightly decreasing agreement for Cosine Similarity and BERTopic. This suggests that Copilot was more aligned with human ratings for BM25 compared to the other models.

Pairwise comparisons between Copilot and individual engineers revealed variation in alignment. Copilot showed the highest agreement with Engineer 1 ($\alpha = 0.58$), and the lowest with Engineer 3 ($\alpha = 0.40$). This reinforces the notion that Copilot tends to align more closely with lenient raters, particularly in cases involving borderline relevance.

Additionally, Figure 5.7 presents the relationship between Copilot and average human ratings. Pearson's correlation coefficient was $r = 0.609$ and Spearman's rank correlation $\rho = 0.614$, indicating a moderate-to-strong association. Notably, green data points – representing instances where all three engineers agreed – cluster along the diagonal, whereas blue and orange points (indicating partial or no agreement) show greater deviation. This suggests that Copilot's ratings are more reliable in cases where human consensus is high.

[3]The standard interpretation scale for Krippendorff's Alpha is: $\alpha \geq 0.80$ = strong agreement; $0.67 \leq \alpha < 0.80$ = substantial or tentative agreement; $0.40 \leq \alpha < 0.67$ = moderate agreement; $0.20 \leq \alpha < 0.40$ = slight agreement; $\alpha < 0.20$ = poor or no agreement [26].

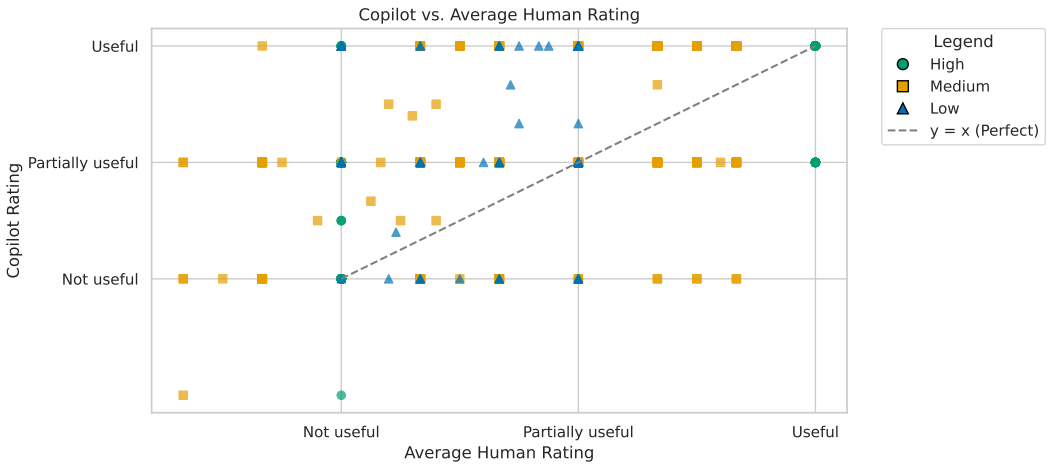


Figure 5.7: Comparison of Copilot ratings versus the average of engineer ratings for the same associations. Each point represents one association, with marker shape and colour indicating the prediction rank group: high (circle), medium (square), and low (triangle). The diagonal $y = x$ line indicates perfect agreement between Copilot and human ratings.

Prediction Rank and Relevance

Beyond overall scores, we also examined how the position of each prediction in the ranked list correlates with its perceived usefulness. Figure 5.8 and Table 5.2 collectively illustrate a strong relationship between the rank position of a predicted association and its perceived usefulness, as rated by both engineers and Copilot. Ratings range from *not enough information* to *useful*, and a clear downward trend in perceived relevance is observed as the rank increases.

The top-ranked predictions (Rank 1) were rated as *useful* in 62.5% of cases, with relatively few *not useful* (16.9%) or *not enough information* (2.7%) responses. This suggests that the first suggestion returned by a model is frequently judged as highly relevant. However, as rank position increases, perceived usefulness consistently declines. By Rank 5, only 20.5% of suggestions were labelled as *useful*, while *not useful* ratings rose to over 51%.

This trend is consistently observed across all human evaluators, as shown in Figure 5.8, where average rating scores decrease sharply from Rank 1 to Rank 5. Copilot exhibits a slightly more favourable rating pattern but follows the same declining trajectory, indicating a similar sensitivity to rank position.

A more detailed breakdown of rating distributions per rank is provided in Appendix B Table B.1, which reports position-wise relevance scores for each model. These findings further validate the effectiveness of rank-based prioritisation across all retrieval approaches.

Average Ratings

Average ratings per model and evaluator, summarised in Figure 5.9, reveal consistent patterns in perceived usefulness across the engineers and the Copilot. Copilot provides high

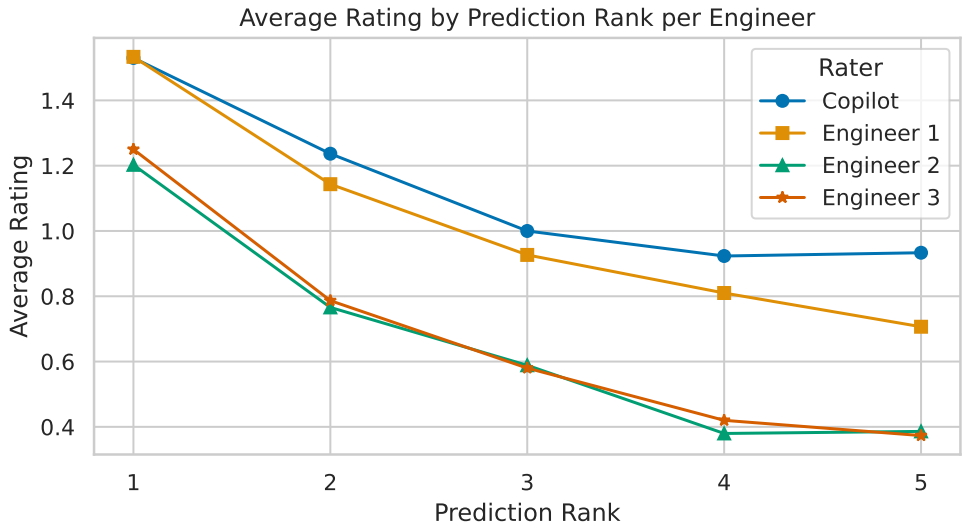


Figure 5.8: Average rating of top-5 predictions per evaluator, based on the scale: 0 = Not useful, 1 = Partially useful, 2 = Useful. Usefulness tends to decrease with lower-ranked predictions.

	Not enough info	Not useful	Partially useful	Useful
Prediction 1	2.7%	16.9%	17.9%	62.5%
Prediction 2	2.4%	34.3%	23.1%	40.3%
Prediction 3	3.1%	42.6%	25.5%	28.7%
Prediction 4	2.8%	50.3%	24.2%	22.7%
Prediction 5	3.4%	51.1%	24.9%	20.5%

Table 5.2: Distribution of usefulness ratings across prediction ranks 1 to 5. Higher-ranked predictions tend to receive higher proportions of *useful* ratings, with a notable decline in usefulness and an increase in *not useful* ratings at lower ranks.

and stable scores across all three models, ranging from 1.09 to 1.15, with a slight preference for Cosine Similarity (1.15). Among the human evaluators, Engineer 1 and Engineer 2 both rate BM25 the highest, with average scores of 1.13 and 0.74, respectively. This suggests a shared preference for classical retrieval techniques. In contrast, Engineer 3 assigns the highest score to Cosine Similarity (0.75), reflecting different retrieval expectations or subjective evaluation tendencies. These inter-rater variations underscore the diversity of judgment criteria, even within a standardised annotation task.

However, it is worth noting that even the highest average scores from the engineers remain relatively low on the scale from -1 to 2, where -1 corresponds to *not enough information*, 0 corresponds to *not useful*, 1 to *partially useful*, and 2 to *useful*. For instance, Engineer 2's highest score of 0.74 still falls below the *partially useful* midpoint, indicating that most suggestions were judged to be of limited practical relevance.

Comparison Type	Description	Kruskal-Wallis p-value
Between models (Engineer 1)	Individual model ratings by Engineer 1	0.0029^{**}
Between models (Engineer 2)	Individual model ratings by Engineer 2	0.2274
Between models (Engineer 3)	Individual model ratings by Engineer 3	0.0358^{*}
Between models (Copilot)	Model ratings assigned by Copilot	0.4580
Between evaluators (BM25)	Ratings of BM25 by all evaluators	$< 0.001^{***}$
Between evaluators (Cosine Similarity)	Ratings of Cosine Similarity by all evaluators	$< 0.001^{***}$
Between evaluators (BERTopic)	Ratings of BERTopic by all evaluators	$< 0.001^{***}$
Between models (All engineers, excl. Copilot)	Aggregated model ratings (Eng. 1–3)	0.0639

Table 5.3: Summary of Kruskal–Wallis tests assessing differences in model ratings across evaluators and vice versa. Significance levels: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.

To assess the statistical significance of the differences in the ratings, we conducted Kruskal-Wallis tests – a non-parametric alternative to one-way ANOVA – appropriate for our ordinal, non-normally distributed 4-point rating scale. The tests were applied in two settings:

1. Comparing average ratings across models *within each evaluator*.
2. Comparing average ratings across evaluators *within each model*.

A summary of all Kruskal–Wallis test results is provided in Table 5.3. Results indicate statistically significant differences in model ratings for Engineer 1 ($p = 0.0029$) and Engineer 3 ($p = 0.036$), but no significant differences for Engineer 2 ($p = 0.227$) or Copilot ($p = 0.458$). These findings suggest that Engineers 1 and 3 exhibit stronger model preferences, whereas Engineer 2 and Copilot provide more uniform ratings across models.
When examining inter-evaluator differences for each model, the Kruskal-Wallis test yielded $p < 0.001$ for all three models, indicating highly significant variability across human raters. This reinforces earlier observations of evaluator-specific biases and highlights the importance of accounting for such variation in downstream analyses.
Finally, we examined overall model performance by aggregating scores across all human evaluators. The mean scores for BM25, Cosine Similarity, and BERTopic were 0.85, 0.83, and 0.76, respectively. The Kruskal-Wallis test yielded $p = 0.064$, indicating that although BM25 shows a modest performance advantage, the differences among models are not statistically significant at the conventional 0.05 threshold.

Agreement Patterns and Edge Cases

Agreement patterns between engineers and between engineers and Copilot offer critical insights into the consistency and reliability of the annotation process. Out of the 790 associations evaluated by all three engineers, full consensus, where all three assigned the same label, was reached in 305 cases. An additional 345 associations showed partial consensus, with two engineers agreeing and one differing. This results in 650 out of 790 cases (82.3%) exhibiting majority agreement, suggesting strong overall alignment among human evaluators. In the subset evaluated by only two engineers, consensus was observed in 434 out of 710 cases.

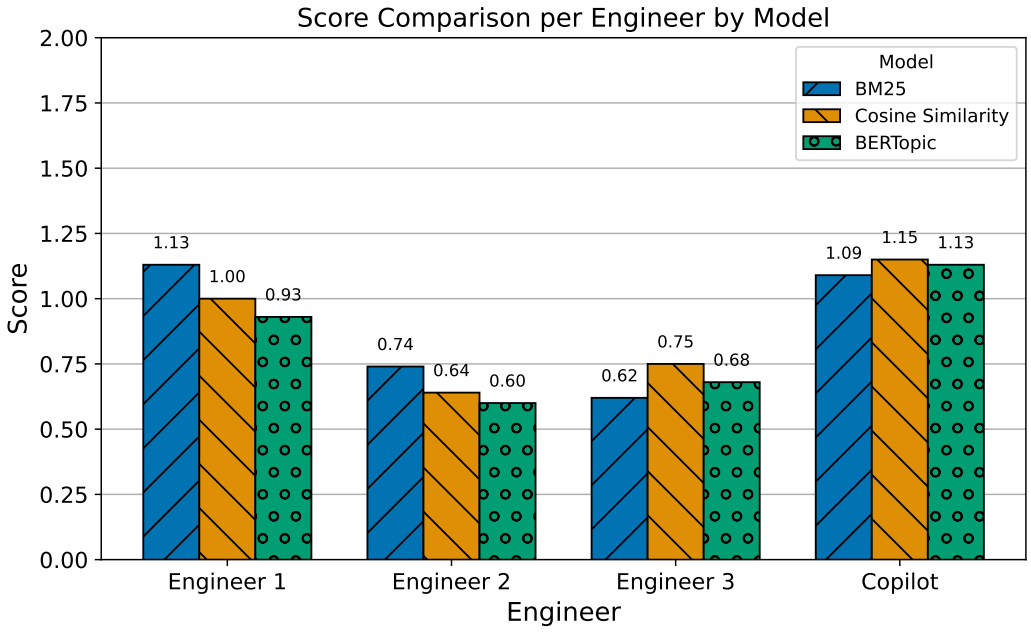


Figure 5.9: Average usefulness scores assigned by each engineer and Copilot for the three evaluated models. Higher scores indicate a higher perceived usefulness of the model's outputs.

Table 5.4 details the number of consensus cases per rating across models. Notably, most human agreement occurred at the lower end of the scale, particularly for rating *not useful*, which had the highest number of consensus cases: 173 for BM25, 180 for Cosine Similarity, and 213 for BERTopic. In contrast, consensus for rating *useful* was consistently lower, indicating that judgments of usefulness are more subjective and context-dependent, while judgments of irrelevance are more consistent.

Regarding Copilot's alignment with human evaluators, the model fully agreed with all three engineers in 225 cases overall (BM25: 81, Cosine Similarity: 72, BERTopic: 72). In cases with only two human annotations, Copilot matched both in 267 instances (BM25: 77, Cosine Similarity: 97, BERTopic: 93). Moreover, Copilot agreed with majority of engineers in 653 cases. However, complete disagreement between Copilot and all three human raters occurred in 186 cases (BM25: 64, Cosine Similarity: 57, BERTopic: 65). Similarly, in the two-annotator setting, Copilot assigned a different label than both engineers in 250 instances (BM25: 80, Cosine Similarity: 75, BERTopic: 95). These results indicate that while Copilot often aligns with human consensus, significant discrepancies still persist.

A closer analysis of rating divergence reveals further nuance. In 148 of the 790 cases, ratings ranged from *not useful* to *useful*, reflecting strong disagreement. Interestingly, 72 cases contained all three ratings, illustrating maximal subjectivity. In the two-rater subset, 106 samples had disagreements between *not useful* and *useful*. These distributions suggest that Copilot is more likely to diverge when human raters themselves exhibit disagreement,

	Not enough info	Not useful	Partially useful	Useful
BM25	3	173	37	139
Cosine Similarity	4	180	40	132
BERTopic	3	213	39	121

Table 5.4: Distribution of majority agreement ratings for each model. Each cell indicates the number of predictions where the majority of engineers assigned the same usefulness rating.

especially when relevance is ambiguous.

As further illustrated in Figure 5.7, Copilot's agreement with human evaluators improves as the level of human consensus increases, reinforcing the importance of rater alignment in evaluating model reliability.

Prediction Overlap and Model Output Diversity

To assess the degree of content overlap among the three association discovery models, we analysed the top-5 ranked outputs produced for each issue and computed both the absolute overlap in predictions and Jaccard similarity scores. This analysis helps determine whether the models tend to retrieve the same associations or identify distinct candidates, which is critical for understanding redundancy and complementarity across methods.

Among the evaluated methods, BM25 generated the largest number of unique suggestions, with 313 associations retrieved exclusively by this model. In comparison, Cosine Similarity and BERTopic produced 107 and 129 unique associations, respectively. These figures suggest that BM25 explores a broader or more distinct semantic space, while the other two models have greater overlap.

Pairwise comparison of model outputs reveals further insights into shared retrieval behaviour. BM25 and Cosine Similarity overlapped on 173 associations, while BM25 and BERTopic shared 151. The highest overlap was observed between Cosine Similarity and BERTopic, which shared 357 predictions. Additionally, 137 associations were retrieved by all three models, forming the core consensus subset among retrieval strategies. The distribution of overlaps is visualised in Figure 5.10.

Jaccard similarity scores reinforce these findings. The strongest similarity is between Cosine Similarity and BERTopic ($J = 0.555$)[4], consistent with their shared reliance on SBERT embeddings. In contrast, BM25 demonstrates lower similarity with both Cosine Similarity ($J = 0.209$) and BERTopic ($J = 0.178$), indicating that BM25 contributes greater diversity to the overall candidate pool.

Table 5.5 presents evaluator ratings for the 137 associations retrieved by all three models. This subset, representing the most consistently identified suggestions, received notably higher usefulness ratings across evaluators. Engineer 1 labelled 81.5% of these associations

[4]Jaccard similarity ranges from 0 to 1, where 0 indicates no overlap and 1 indicates complete overlap between two sets. In this context, higher scores imply greater overlap in the top-5 predicted associations.

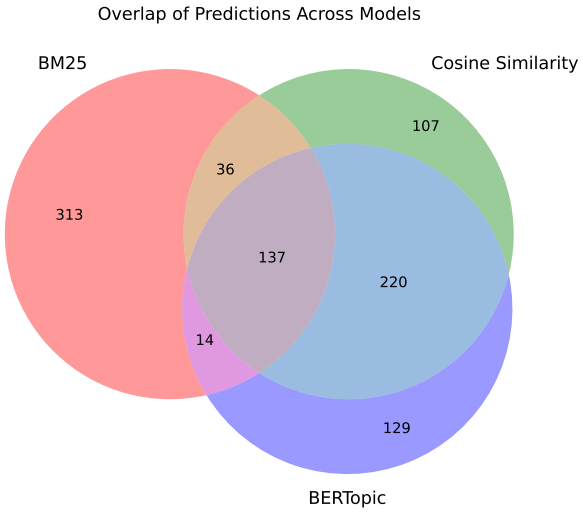


Figure 5.10: Overlap of predicted associations generated by BM25, Cosine Similarity, and BERTopic retrieval models. The diagram illustrates how many predictions are unique to each model and how many are shared across models.

	Not enough info	Not useful	Partially useful	Useful
Engineer 1	1.9%	7.1%	9.5%	81.5%
Engineer 2	7.9%	5.8%	18.6%	67.7%
Engineer 3	0%	25.8%	13.6%	60.6%
Copilot	0%	6.6%	29.2%	64.2%

Table 5.5: Distribution of evaluator ratings for the 137 associations retrieved by all three models. This subset received higher usefulness ratings across all evaluators.

as *useful*, while Engineer 3 – typically the strictest – assigned *useful* ratings in 60.6% of the cases. Copilot rated 64.2% of these associations as *useful*.

Figure 5.11 provides a visual comparison between the usefulness ratings for this overlapping subset and the full set of evaluated predictions. The group of overlapping associations (blue bars) received a much higher percentage of *useful* ratings (69.9%) compared to the overall pool (34.0%), while also showing a sharp decrease in *not useful* ratings. These results indicate that inter-model consensus is positively correlated with perceived utility. While the comparison is not perfectly balanced – 137 overlapping associations are contrasted against over 1,500 overall predictions – the observed trend remains meaningful. The sharp contrast in ratings supports the idea that predictions identified by multiple methods are more likely to be judged as useful by human experts.

This finding supports the potential of hybrid or ensemble approaches that prioritise associations identified by multiple methods. Incorporating such strategies may enhance the quality and reliability of semantic association recommendations in industrial engineering contexts. Taken together, these results suggest that while Cosine Similarity and BERTopic tend to

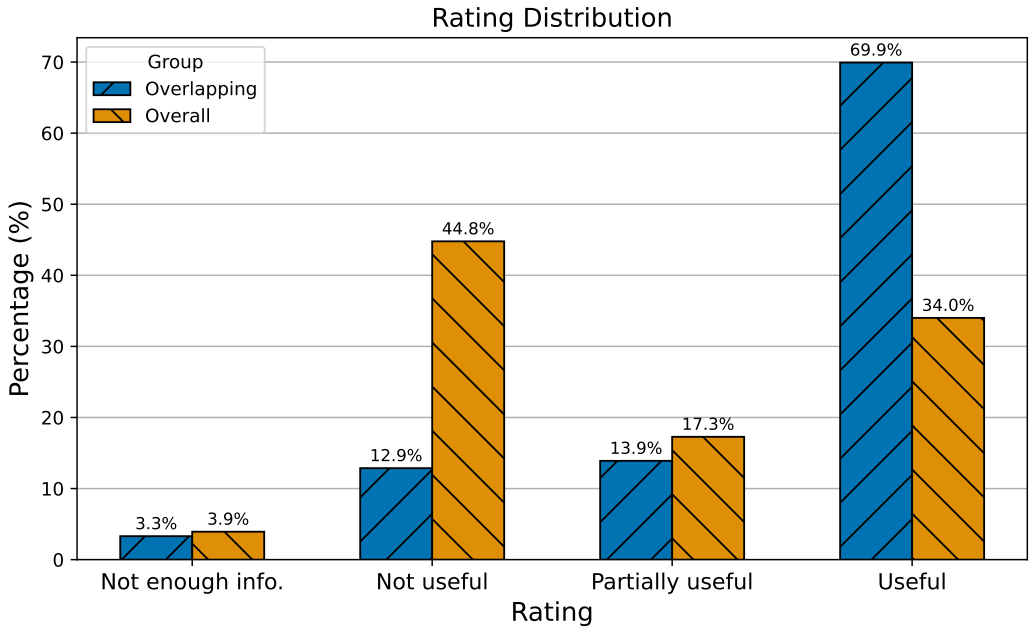


Figure 5.11: Evaluator rating distribution for all evaluated associations (Overall) vs. associations retrieved by all three models (Overlapping).

identify similar associations, BM25 offers a complementary perspective by surfacing distinct candidates. This diversity in model outputs highlights the potential for ensemble or hybrid approaches that integrate retrieval strategies to improve coverage and robustness in automated association discovery.

5.4 Discussion

5.4.1 Retrieval Performance and Model Trade-Offs

Among the implemented models, **BM25** emerged as the top-performing method in the quantitative evaluation, successfully retrieving 3,277 of the 6,378 ground-truth associations (Hit@10 = 51.4%). This performance is notable considering BM25's simplicity and purely lexical nature. Its strength likely stems from how engineering issue descriptions are written: with relatively consistent and domain-specific terminology. In many cases, engineers use standard phrases or repeat specific keywords when describing recurring problems, component names, or root causes. BM25 is well-suited to capture this kind of frequency-weighted term overlap, giving it a practical edge in contexts where vocabulary repetition is common and meaningful.

In contrast, embedding-based models such as **SBERT + Cosine Similarity** (48.1%) and **SBERT + BERTopic** (47.6%) encode richer semantic information, capturing latent similarities between descriptions with different wording. However, their advantage over BM25

was marginal. One reason may be that engineering texts, though noisy, tend to use fixed expressions and acronyms, which embeddings may underrepresent or average out. Moreover, domain-specific terms frequently fall outside the distribution of the training data used by general-purpose embedding models. When such terms are unseen or rare in the model's training corpus, their embeddings may not accurately reflect their technical meaning – unlike lexical models such as BM25, which operate purely on term overlap and are unaffected by semantics or vocabulary mismatch. Additionally, while SBERT models excel at capturing general semantic similarity, they may struggle to distinguish fine-grained technical details.

Clustering-based models like **TF-IDF + DBSCAN** (47.1%) and **TF-IDF + k-Means** (42.4%) attempted to group semantically similar issues before narrowing the candidate pool, but results were inconsistent. Clustering may help reduce noise and improve precision when issue descriptions are long and rich in context. However, many engineering tickets are short or sparsely worded, which can lead to brittle cluster boundaries and poor topic separation. Methods using **Word2Vec**, **FastText**, and **LDA** performed worst overall (mostly under 41%), likely due to their reliance on either static embeddings or bag-of-words assumptions that do not capture context or syntax well.

Furthermore, a detailed post-hoc analysis revealed that 2,077 of the 6,378 ground-truth associations – approximately one-third – were **procedural rather than semantic** in nature. These links were typically added by engineers for workflow continuity, such as follow-up actions, test coordination, or documentation references. While such links are meaningful in operational contexts, they do not reflect true semantic similarity between issue descriptions and thus lie outside the scope of this study. From the perspective of semantic association discovery, they function as label noise, undermining the reliability of the evaluation and lowering the apparent performance of content-based methods.

This limitation exposes a critical tension in the task design: while models are evaluated on their ability to retrieve ground-truth associations, a substantial portion of those associations are not semantically justified. Consequently, true model performance on the intended task, identifying meaningful textual similarity, is likely higher than quantitative metrics suggest.

In parallel, the qualitative evaluation further underscores the distinction between semantic similarity and practical relevance. Even among top-ranked predictions with high similarity scores, human experts often rated suggestions as only partially useful or not useful at all. This reveals a second challenge: **semantic proximity at the text level does not guarantee contextual or functional usefulness in engineering workflows**. Descriptions that share vocabulary or phrasing may still differ in purpose, context, or technical scope, limiting their value for knowledge reuse.

5.4.2 Variability in Human Judgment and Evaluation Challenges

The evaluation revealed substantial inter-rater variability among the three participating engineers. Engineer 1 consistently assigned higher scores, labelling 46% of associations as *useful*, while Engineer 3 applied stricter criteria, rating over half of the suggestions as *not useful*. Engineer 2 showed a more balanced rating distribution but evaluated only a subset of the dataset (790 out of 1,500 predictions), which limits comparability and introduces

potential sampling bias in the aggregated scores and agreement calculations. This uneven coverage, combined with differing annotation styles, contributed to the moderate inter-rater agreement (Krippendorff's $\alpha \approx 0.53$), reflecting the inherently subjective and nuanced nature of judging semantic similarity in technical texts.
Several factors contribute to this variability. First, engineers differ in how they interpret *usefulness*. Second, the level of domain familiarity and recent exposure to specific issue types likely shapes judgment. Engineers with deep experience in a particular subsystem may spot meaningful links that are not apparent from surface-level text alone, while others may exercise caution in the absence of explicit signals. Third, the annotation task is cognitively demanding: evaluating 500 associations across three different models requires sustained focus and consistent decision-making, which can vary across individuals and over time. The lack of a shared annotation rubric or calibration session further amplifies these effects.
Notably, consensus was more frequent on clearly *unhelpful* associations (rating 0), whereas positive judgments were more varied. This suggests that engineers share a consistent understanding of *irrelevance*, but differ in their thresholds for what constitutes a *useful* or *partially useful* recommendation – likely influenced by personal experience, expectations, and project-specific context.
Direct feedback from the evaluators further illustrates the sources of disagreement and practical challenges they encountered:

- **Semantic Disconnection and Dataset Diversity:** Engineers noted that many issue reports were not semantically related, in part due to the intentional inclusion of a wide range of projects and systems. While this diversity was necessary to ensure representativeness and test the scalability of association methods, it also made it more difficult to apply consistent relevance criteria and contributed to inter-rater disagreement.
- **Sufficiency of Title and Description:** Despite the evaluation challenges, annotators agreed that the `Title` and `Description` fields were generally sufficient for judging semantic relevance. Structured fields such as component labels or keywords were not necessary. However, one annotator noted that including basic contextual cues, such as the project or system name, could further aid interpretation, especially in cases involving ambiguous terminology or cross-domain issues. This reinforces the value of focusing on unstructured text while acknowledging the potential benefit of minimal contextual metadata.
- **Ambiguity in Entity References:** Annotators observed that the models failed to distinguish between identical terms referring to different entities (e.g., "Stingray" used both as a project name and a tool). The issue reflects a limitation in the models' ability to resolve entity ambiguity from context, highlighting the potential value of incorporating disambiguation mechanisms.

These findings highlight the complexity of evaluating semantic associations in industrial issue tracking. While human judgment offers valuable domain expertise, it is inherently subjective and varies across annotators. Differences in expectations, domain familiarity, and interpretation of usefulness contribute to inconsistent ratings. This variability underscores the need for complementary evaluation strategies that can provide scalable, consistent, and

repeatable assessments. To address this, we introduced a large language model (Copilot) as an additional evaluator. The following sections examine how Copilot's ratings compare to human annotations, and whether it can serve as a reliable proxy for expert judgment in semantic relevance evaluation.

Evaluation Patterns Across Human Annotators and Copilot

The differences in how Copilot and the human annotators rated the same set of associations reveal meaningful contrasts in evaluation style and underlying reasoning. Compared to the engineers – particularly the stricter Engineer 3 – Copilot showed a marked tendency to avoid extreme judgments. It assigned significantly fewer *not useful* labels and almost never selected *not enough information*, instead clustering its responses around the middle category of *maybe useful*. This middle-ground behaviour suggests that Copilot is more tolerant of weak or ambiguous associations, where a human might apply a stricter threshold for usefulness or decline to rate altogether.

This moderation likely stems from its nature as a general-purpose language model: trained to predict plausible continuations and identify semantic patterns, rather than to make high-stakes, domain-specific relevance decisions. Without awareness of the practical implications of surfacing irrelevant associations, such as wasted engineer time or confusion, Copilot leans toward inclusive scoring, often treating surface-level semantic similarity as sufficient. By contrast, engineers are less forgiving of superficial overlaps and more aware of subtle contextual mismatches that can render two issues unrelated despite sharing terminology.

Another notable behaviour is Copilot's reluctance to express uncertainty. While Engineer 2 made liberal use of the *not enough information* label in cases of ambiguity or underspecified context, Copilot almost never used it. This suggests a form of overconfidence or a limitation in the model's ability to recognise when it lacks sufficient information. Prior research has documented similar behaviour in LLMs, demonstrating that they often produce high-confidence judgments even when incorrect, a known challenge in calibrating large language models [19]. Without explicit prompting to reason about uncertainty or defer judgment, Copilot tends to assign confident scores, even when a human would hesitate.

One possible factor contributing to Copilot's relative leniency is the absence of task-specific optimisation, constraints or cost function for over-inclusion. Unlike human annotators, who may consciously avoid recommending irrelevant associations due to their potential to mislead users, Copilot was not explicitly trained for this evaluation task and likely lacks a domain-specific penalty for over-inclusion. As a result, it appears more inclined to assign partial credit in ambiguous cases, where an expert might apply stricter criteria. This tendency aligns it more closely with the most permissive human rater (Engineer 1), and contributes to its divergence from more conservative evaluators like Engineer 3. However, given the limited transparency into Copilot's training data and objectives, these interpretations remain speculative and should be treated as empirical observations rather than definitive explanations.

These patterns highlight both the promise and limitations of using LLMs in semantic association evaluation. On one hand, Copilot's consistency and balanced rating distribution make it a potentially useful tool for scaling annotation tasks or acting as a "second opinion"

in ambiguous cases. On the other hand, its lack of domain sensitivity, cautious judgment, and ability to express uncertainty limits its reliability as a standalone evaluator, particularly in high-precision or safety-critical applications.

For future use, Copilot may still offer value as a scalable screening tool. Its tendency to avoid strong rejections and hedge toward partial relevance makes it suitable for narrowing down large pools of candidate associations before expert review. In this hybrid workflow, Copilot could reduce human workload while deferring final judgment to domain experts. Enhancing its utility may involve domain-specific prompt tuning, integrating confidence estimation, or exposing it to task-specific training examples that teach the distinction between truly relevant and merely similar associations.

Ultimately, while Copilot demonstrates some alignment with human annotators, it cannot fully replace expert judgment, especially in contexts that require nuanced, cautious interpretation. Its role is best seen as complementary, not substitutive—offering scalable support for semantic evaluation while preserving the critical role of human expertise in final decision-making.

5.4.3 Model-Specific Evaluation Patterns

The model-specific rating distributions in Figure 5.5 and Figure 5.9 offer a more granular view of how association quality varies across BM25, Cosine Similarity, and BERTopic.

Among human annotators, BM25 received the highest proportion of *useful* ratings (36%), suggesting that lexical matching still offers strong signal strength in this domain. SBERT closely followed, while BERTopic lagged slightly and drew the highest share of *not useful* labels (48%). This indicates that topic-based clustering may introduce more irrelevant or loosely related links, likely due to coarse-grained groupings or overlapping topics that obscure precise semantic boundaries.

The differences across models are meaningful: even though absolute performance gaps are not large, consistent trends across annotators suggest that BM25 was more reliably aligned with engineer expectations. SBERT's comparable performance reflects its ability to capture broader semantic similarity, but also highlights the challenge of distinguishing useful from merely related issues. BERTopic's lower ratings may stem from instability in topic granularity or insufficient discrimination within technical vocabulary.

In contrast, Copilot's ratings (shown in Figure 5.9) reveal less variation across models. Its evaluations were dominated by *maybe useful* labels, with relatively small differences in *useful* or *not useful* proportions between methods. This flattening effect suggests that Copilot is less sensitive to method-specific quality differences. As a result, it may be less reliable for comparing models directly, especially in tasks where subtle distinctions in relevance matter.

Overall, these findings reaffirm that:

- BM25 outperforms other models in human judgment, despite its simplicity.
- BERTopic underperforms, likely due to limitations in topic coherence or specificity.
- Copilot does not clearly differentiate between models, which limits its utility as a benchmarking tool unless paired with stricter evaluative criteria.

5.4.4 Inter-Rater Agreement and Copilot Alignment

The agreement scores reported in this section provide a more rigorous, quantitative perspective on the variability in annotation behavior discussed earlier. The moderate inter-rater agreement among engineers ($\alpha = 0.53$) confirms that semantic relevance judgments are inherently subjective, even among domain experts. That Copilot's inclusion slightly lowers this agreement ($\alpha = 0.51$) reflects its partial misalignment with stricter human evaluators, particularly Engineer 3.

Interestingly, BERTopic achieved the highest agreement among the models, despite receiving lower absolute usefulness ratings. This suggests that while annotators more consistently agreed on its predictions, often labelling them as irrelevant, the model failed to generate high-value associations. In contrast, BM25 showed the lowest human agreement, indicating that while it retrieved many relevant items, the judgments were more contentious, perhaps due to its inclusion of borderline or superficial matches.

Copilot's agreement pattern reinforces its alignment with more lenient evaluators, particularly Engineer1 ($\alpha = 0.5805$), while diverging most from the strictest rater, Engineer3 ($\alpha = 0.4017$). This aligns with previous findings on Copilot's moderation and inclusivity.

The correlation analysis (Figure 5.7) adds a valuable dimension. With Pearson $r = 0.609$ and Spearman $\rho = 0.614$, Copilot shows moderate-to-strong alignment with average human ratings. Crucially, agreement strengthens when human consensus is high (green points clustering along the diagonal). This implies that Copilot is most trustworthy when humans agree, but less reliable in ambiguous or contentious cases.

These insights underscore a practical takeaway: Copilot is not a replacement for expert judgment, but it can serve as a scalable proxy in cases of clear semantic overlap. Where human disagreement is high, its reliability diminishes, highlighting the need for human oversight or hybrid evaluation strategies.

5.4.5 Prediction Rank as an Indicator of Semantic Relevance

The consistent decline in perceived usefulness from Rank 1 to Rank 5 across both engineers and Copilot evaluations highlights that the ranking order produced by retrieval models carries meaningful semantic information. However, the steep drop-off, particularly after the top two positions, also reveals the difficulty of maintaining relevance further into the ranking. While over 60% of top-ranked predictions were rated as *useful*, by Rank 3 this had dropped below 30%, with more than half of Rank 5 suggestions rated *not useful*. This sharp gradient suggests that most models can confidently identify one or two strong candidates, but struggle to reliably extend relevance further, likely due to limited semantic signals, noisy text, or embedding drift. This drop in performance is often linked to how retrieval models operate: top results typically benefit from clear lexical overlap or strong contextual alignment, but lower-ranked suggestions rely on weaker or more generic signals. In embedding-based approaches, especially those that average representations over long descriptions, important task-specific cues may be overshadowed by less relevant content, reducing the effectiveness of deeper results. The fact that Copilot mirrored this trend, despite its more moderate scoring style, further supports the idea that rank order reflects semantic proximity. Yet, Copilot's

slightly more favourable view of lower-ranked predictions also reinforces its leniency bias discussed earlier.

5.4.6 Interpreting Average Ratings and Statistical Variation

The average rating analysis reveals key differences in how semantic relevance is judged. Copilot scored consistently across models (range 1.09–1.15) and rarely gave low ratings, suggesting a high-recall, low-penalty approach. While this consistency may be useful in coverage-focused tasks, it risks overlooking meaningful differences between models.

Human evaluators, by contrast, provided more varied and generally lower ratings – especially Engineers 2 and 3, whose scores were consistently below the *partially useful* midpoint. Engineer 1 favoured BM25, while Engineer 3 preferred Cosine Similarity, reflecting subjective differences in what each considered *useful*. However, strong disagreement between raters ($p < 0.001$ across models) and Engineer 2's lack of preference highlight the need to involve multiple experts to reduce individual bias and strengthen evaluation robustness. This can be achieved through majority voting, consensus scoring, or weighting areas of agreement more heavily.

Although model differences were not statistically significant ($p = 0.064$), BM25 showed a slight edge in average usefulness. In practical settings, such small advantages – especially when paired with interpretability or speed – can still justify model selection.

Overall, these results highlight a core tension: automated tools offer consistency, while human experts offer nuance. Effective evaluation may require combining both, especially when model deployment involves subjective or domain-specific relevance judgments.

5.4.7 Agreement Patterns and Edge Cases

Agreement patterns reveal distinct dynamics in how humans and Copilot judge semantic associations. Among 790 triply-annotated associations, 82.3% reached majority agreement, primarily on *not useful* labels, highlighting a shared human intuition for rejecting irrelevant associations. This likely stems from negative signals being more salient: engineers confidently dismiss links that span unrelated domains, terminologies, or intent.

In contrast, consensus on *useful* associations was lower, with frequent disagreement across all three labels. Positive judgments demand deeper contextual interpretation, shaped by each annotator's expectations and understanding of utility. This highlights the inherently subjective nature of semantic relevance, especially in borderline cases where surface similarity alone is insufficient.

Copilot matched human ratings well in clear-cut cases but diverged often when engineers disagreed. This suggests that while Copilot can capture general similarity, it lacks the contextual understanding needed for more nuanced judgments. Its disagreement in these cases is not necessarily a mistake – it can serve as a signal that an association is ambiguous and may require further review.

The frequent occurrence of partial agreement between engineers also highlights the value of majority voting in evaluation. High-agreement cases provide reliable supervision, while low-agreement examples expose areas of uncertainty that are useful for model improvement.

Overall, disagreement should not be seen as noise, but as a meaningful signal. Evaluation pipelines should include ways to detect and learn from these edge cases to build more robust and adaptable systems.

5.4.8 Model Diversity, Overlap, and Complementarity

The overlap analysis reveals how the three models contribute differently to semantic association, shedding light on whether they offer complementary value or redundancy. BM25 produced 313 unique associations – far more than Cosine Similarity (107) or BERTopic (129). This reflects its reliance on lexical matching, which allows it to retrieve relevant items that embedding-based models may overlook. Its lower overlap with SBERT-based methods (Jaccard < 0.21) highlights its role in capturing vocabulary-driven associations, including rare terms or edge cases. Cosine Similarity and BERTopic, both SBERT-based, shared over 350 predictions ($J = 0.555$). Despite different mechanisms – direct similarity vs. topic clustering – their shared embedding space leads to similar outputs.

The 137 associations retrieved by all three models received notably higher usefulness ratings across all evaluators. This suggests that when diverse retrieval strategies converge – lexical, semantic, and topical – the result is more robust and contextually appropriate. Such overlap can serve as a reliable signal for prioritising recommendations. While this subset represents only a fraction of the total predictions, its consistently strong ratings across multiple raters support its value as a high-confidence zone. These cases may not cover the full diversity of issue types, but they offer a practical foundation for trust-based filtering or model calibration in operational deployments. These findings support the use of hybrid retrieval pipelines that exploit both overlap (for precision) and diversity (for coverage). Consensus predictions can be elevated as "high confidence" outputs, while model-specific results expand discovery. In practical settings, this balance improves both trust and utility in recommendation systems.

5.4.9 Conclusion and Answer to the Research Question

RQ: How can semantic association techniques be used to automatically recommend related engineering issues in real-world settings?

This project shows that automated semantic association techniques can be used to recommend related issues, especially when grounded in high-quality text representations and aligned with domain-specific evaluation. Across multiple retrieval strategies, we find that models like BM25, Cosine Similarity, and BERTopic each offer distinct advantages – lexical coverage, semantic closeness, and topic coherence, respectively. Their consensus strongly correlates with human-judged usefulness, suggesting ensemble approaches can improve reliability.

However, the task is inherently subjective. Human evaluators disagreed often – especially on borderline or partially relevant cases – highlighting that context, expectations, and user intent all shape the perception of "usefulness." While Copilot offers stable and high-recall assessments, its divergence on edge cases suggests it is best used as a supporting tool, not a replacement for expert judgment.

Effective deployment of association systems thus requires hybrid evaluation strategies, careful integration of diverse retrieval signals, and mechanisms to flag low-confidence or disputed cases. This work offers practical insights into building such pipelines, showing how human-in-the-loop feedback, model consensus, and evaluator agreement patterns can jointly improve semantic recommendation quality in industrial issue tracking.

Chapter 6

Conclusions and Future Work

This chapter summarises the key contributions of the thesis, reflects on the main findings in relation to the research questions, and highlights the broader implications of the work. It concludes with a discussion of identified limitations and outlines directions for future research and development.

6.1 Conclusion

This thesis examined the application of Natural Language Processing and Machine Learning techniques to support the classification and association of engineering issue reports. In response to the main research question – **how AI-driven techniques can be leveraged to automate and enhance the classification and association of industrial issue reports** – this work implemented and systematically evaluated a wide range of models under real-world constraints.

RQ1 examined how **multi-label classification** techniques can be effectively applied to categorise industrial issue reports across a large and diverse set of categories. Over 70 model configurations were evaluated on 23,534 issues across 30 categories. Transformer-based models, particularly **DistilBERT**, demonstrated strong top-k performance (Recall@5 = 0.93, Recall@7 = 0.96), despite data sparsity and class imbalance. Classical models such as **TF-IDF (n-grams) + Logistic Regression (OvR)** offered a competitive alternative (Recall@5 = 0.84, Recall@7 = 0.93), with advantages in deployment and interpretability. These findings confirm that while deep models offer superior predictive accuracy, classical pipelines remain valuable, especially where transparency or resource efficiency is critical.

RQ2 addressed the challenge of **semantic association discovery**, focusing on how similar or related issues can be identified automatically. **Lexical (BM25)**, **semantic (SBERT + cosine similarity)**, and **topic-based (BERTopic)** methods produced overlapping top-5 recommendations, and the associations identified by these approaches were rated as useful in over 70% of the cases by engineers. Interestingly, this suggests that agreement between fundamentally different retrieval methods may serve as a proxy for relevance. The inclusion of Copilot as an automated evaluator showed systematic overestimation of relevance scores compared to expert ratings, and limited agreement in borderline cases. These findings sug-

gest that while LLM-based evaluators can provide consistent judgments, they may not yet be reliable substitutes for domain experts in specialised industrial contexts.
In response to the main research question, the findings demonstrate that AI-based techniques can support more consistent, scalable, and semantically aware issue tracking. While no single method dominated across all dimensions, the combination of classical machine learning, transformer-based classification, lexical-semantic retrieval, and human-in-the-loop evaluation produced actionable results. The comparative analysis across model families, the inclusion of expert evaluation, and the attention to domain-specific constraints contribute to a more grounded understanding of how NLP methods can be applied in engineering issue management.
Several insights emerged from this work. First, the performance of classical models exceeded expectations, especially given the noisy and incomplete labels – suggesting that simple, interpretable methods should not be overlooked in industrial settings, where the frequent mention of specific technical terms or keywords can often provide sufficient signal for effective classification. Second, the human evaluation revealed nuances that metrics alone could not capture, highlighting the importance of incorporating expert feedback even in highly automated pipelines.
For industry, the key lesson is that AI-based classification and retrieval can provide meaningful support even when trained on imperfect data – provided models are carefully selected and evaluated with real-world constraints in mind. For academia, this study provides a grounded case for moving beyond clean benchmark datasets and testing models in applied, noisy, and high-impact environments. Future work will benefit from continuing to bridge this gap between theoretical performance and operational feasibility.

6.2 Threats to Validity

Despite the breadth of experiments and analyses conducted in this thesis, several threats to validity should be acknowledged:

Model and Technical Constraints:

- Computational Constraints: The lack of access to GPUs and scalable cloud infrastructure limited the depth of transformer fine-tuning and hyperparameter optimisation. This constraint was accepted due to company security policies and time restrictions. To mitigate its effect, lightweight models were selected, and training pipelines were optimised for local execution. Despite these limits, strong performance was achieved, showing that even constrained transformer variants can be viable in practical industrial contexts.
- Lack of Access to Proprietary LLM APIs: Due to strict network restrictions and data privacy policies in the organisation, the study could not utilise proprietary large language models for inference or training. This excluded the use of advanced LLM-based techniques, including zero-shot or few-shot prompting and fine-tuning on internal issue report data, methods that are increasingly prominent in academic and applied NLP research. The exclusion was necessary to comply with industrial confidentiality standards and prevent any external data transmission. As a result, the study

focused on open-source and locally deployable models to ensure both reproducibility and data security within the existing engineering infrastructure. To still investigate the potential role of LLMs in the issue tracking pipeline, Copilot was experimentally introduced as an automated evaluator, rather than as a modelling approach. Specifically, Copilot was tasked with assessing the usefulness of model-predicted associations, acting as a proxy for human judgment. This allowed for a limited, indirect exploration of LLM capabilities without violating data policies, while also revealing challenges around LLM-based evaluation reliability in domain-specific settings.

Data and Evaluation Limitations:

- Restricted Dataset Size, Quality and Coverage: The classification and association tasks were based on a relatively small and imperfectly labelled dataset. For classification, category labels were often incomplete, inconsistently applied, and not fully aligned with how engineers use them in practice. For association, only a limited number of gold-standard links were available. These conditions introduced potential biases in both training and evaluation, particularly where correct but unannotated predictions may have been unfairly penalised. This risk was accepted to preserve the realism of the study, as such noise and sparsity are common in industrial datasets. To mitigate the impact, the evaluation relied on top-k metrics (e.g., Recall@5 and Recall@7), multi-label-aware measures and expert evaluation, which are more appropriate for assessing model performance when ground truth labels are incomplete or partially missing. These constraints may limit the generalizability of the findings to more diverse issue types or other engineering domains.
- Bias in Expert Evaluation: The expert evaluation of associations may be affected by two types of bias: sampling bias, since the evaluated subset may not fully reflect the dataset's diversity (e.g., rare or complex cases); and evaluator bias, as individual interpretations of "usefulness" varied across engineers. These risks were accepted due to resource constraints and the need for domain-specific insight. To reduce their impact, issues were sampled across multiple categories, rated independently by multiple experts, and assessed using inter-rater agreement metrics and complementary automatic evaluations.

Generalizability and External Validity:

- Limited Number of Expert Evaluators: Only three domain experts were available to evaluate the recommended associations. This introduces subjectivity and potential bias in the assessment of relevance and usefulness. The limited number was due to the highly specialised nature of the task: domain-specific engineers with relevant expertise are scarce, and their time is exceptionally valuable. This evaluation was only made possible because dedicated expert time was explicitly allocated to the project, which is not typically feasible at scale. To mitigate the impact of this constraint, inter-rater agreement metrics (e.g., Krippendorff's Alpha) were calculated, and statistical comparisons across models were conducted using overlapping prediction sets and significance testing. While broader validation would strengthen the generalisability

of the findings, the focused evaluation already yielded actionable insights into model performance, practical relevance, and expert-model agreement patterns.
- Limited Generalizability Across Companies and Domains: This study was conducted using internal issue report data from a single organisation which operates within a specific engineering domain and workflow structure. While the dataset is representative of real-world industrial challenges, the findings may not directly generalise to other companies, especially those in different sectors or with alternative issue-tracking systems. The domain-specific vocabulary, reporting style, and annotation practices likely influenced both model performance and evaluation outcomes. This threat to external validity was accepted to maintain focus and depth within a well-scoped industrial context. To mitigate it, the thesis emphasises methods that are modular, interpretable, and adaptable, enabling future studies to explore whether these methods can be adapted and applied effectively in other organisational or industrial contexts.

6.3 Future Work

This study opens several directions for further investigation and system development:

- **Incorporate Real-Time Feedback Loops:** Develop systems that allow engineers to provide in-context feedback on classification or association suggestions, enabling adaptive learning and continuous improvement.
- **Expand and Diversify Human Evaluation:** Engage a broader pool of annotators, potentially across multiple teams or sites, and introduce calibration sessions to align the interpretation of usefulness. Crowdsourcing or expert panel reviews could improve annotation consistency.
- **Explore LLM-based Models:** Future work should test LLM models on both classification and association tasks, leveraging few-shot capabilities and better generalisation for edge cases.
- **Conduct Calibration Studies on Acceptable Performance Thresholds:** While current models demonstrate strong performance on automated metrics, it remains unclear what level of accuracy or recall is sufficient for real-world adoption. Future studies should involve engineers directly to identify the minimum acceptable performance needed to trust and integrate such models into their workflow.
- **Improve Label Quality via Semi-Automated Curation:** The multi-label classification task was limited by noisy and incomplete labels. Future work could explore semi-supervised, active learning or unsupervised strategies where the model highlights uncertain predictions and engineers confirm or correct them. This would improve label quality iteratively while reducing annotation burden.
- **Analyse Failure Cases and Interpretability:** Future work could focus on systematic analysis of high-confidence false positives/negatives to better understand failure modes. This could lead to improvements in preprocessing, category definitions, or model confidence calibration. Model explainability methods could also be integrated to aid engineers' trust.

- **Optimize Model Architectures and Hyperparameters:** The models in this study were trained using default or minimally tuned hyperparameters due to computational constraints. Future work could involve systematic hyperparameter optimization and architecture tuning – for example, adjusting learning rates, batch sizes, regularization, or model depth – to improve performance.

Bibliography

[1] Petar Afric, Davor Vukadin, Marin Silic, and Goran Delac. Empirical study: How issue classification influences software defect prediction. *IEEE access*, 11:11732–11748, 2023.

[2] Wajdi Aljedaani, Yasir Javed, and Mamdouh Alenezi. Lda categorization of security bug reports in chromium projects. In *Proceedings of the 2020 European symposium on software engineering*, pages 154–161, 2020.

[3] Gabriel Aracena, Kyle Luster, Fabio Santos, Igor Steinmacher, and Marco Aurelio Gerosa. Applying large language models to issue classification. In *Proceedings of the Third ACM/IEEE International Workshop on NL-based Software Engineering*, pages 57–60, 2024.

[4] Muhammad Arslan and Christophe Cruz. Business text classification with imbalanced data and moderately large label spaces for digital transformation. *Applied Network Science*, 9(1):11, 2024.

[5] Jasmin Bogatinovski, Ljupčo Todorovski, Sašo Džeroski, and Dragi Kocev. Comprehensive comparative study of multi-label classification methods. *arXiv preprint arXiv:2102.07113*, 2021.

[6] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[7] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[8] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. Large-scale multi-label text classification on eu legislation. *arXiv preprint arXiv:1906.02192*, 2019.

[9] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. In *Journal of Artificial Intelligence Research*, volume 16, pages 321–357, 2002.

[10] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[11] Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. *Noise reduction in speech processing*, pages 1–4, 2009.

[12] Dingsheng Deng. Dbscan clustering algorithm based on density. In *2020 7th international forum on electrical engineering and automation (IFEEA)*, pages 949–953. IEEE, 2020.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[14] Roman Egger and Enes Gokce. Natural language processing (nlp): An introduction: making sense of textual data. In *Applied data science in tourism: Interdisciplinary approaches, methodologies, and applications*, pages 307–334. Springer, 2022.

[15] Haytame Fallah, Patrice Bellot, Emmanuel Bruno, and Elisabeth Murisasco. Adapting transformers for multi-label text classification. In *CIRCLE (Joint Conference of the Information Retrieval Communities in Europe) 2022*, 2022.

[16] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501, 2024.

[17] Matt W Gardner and Stephen R Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[18] Nadia Ghamrawi and Andrew McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200, 2005.

[19] Tobias Groot and Matias Valdenegro-Toro. Overconfidence is key: Verbalized uncertainty evaluation in large language and vision-language models. *arXiv preprint arXiv:2405.02917*, 2024.

[20] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*, 2022.

[21] Meng Han, Hongxin Wu, Zhiqiang Chen, Muhang Li, and Xilong Zhang. A survey of multi-label classification based on supervised and semi-supervised learning. *International Journal of Machine Learning and Cybernetics*, 14(3):697–724, 2023.

[22] GI Ivchenko and SA Honov. On the jaccard similarity test. *Journal of Mathematical Sciences*, 88:789–794, 1998.

[23] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.

[24] Shardrom Johnson, Sherlock Shen, and Yuanchen Liu. Cwpc_biatt: character–word–position combined bilstm-attention for chinese named entity recognition. *Information*, 11(1):45, 2020.

[25] Robert V Krejcie and Daryle W Morgan. Determining sample size for research activities. *Educational and psychological measurement*, 30(3):607–610, 1970.

[26] Klaus Krippendorff. Computing krippendorff's alpha-reliability, 2011.

[27] William H Kruskal and W Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.

[28] Muhammad Laiq. An intelligent tool for classifying issue reports. In *2023 IEEE/ACM 2nd International Workshop on Natural Language-Based Software Engineering (NLBSE)*, pages 13–15. IEEE, 2023.

[29] Michael P LaValley. Logistic regression. *Circulation*, 117(18):2395–2399, 2008.

[30] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S Yu, and Lifang He. A survey on text classification: From traditional to deep learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(2):1–41, 2022.

[31] Zejian Liang, Yunxiang Zhao, Mengyuan Wang, Hong Huang, and Haiwen Xu. Research on the automatic multi-label classification of flight instructor comments based on transformer and graph neural networks. *Aerospace*, 12(5):407, 2025.

[32] Xueqing Liu and Chi Wang. An empirical study on hyperparameter optimization for fine-tuning pre-trained language models. *arXiv preprint arXiv:2106.09204*, 2021.

[33] Ying Liu, Zhenhao Lin, Da Yin, Yanan Gao, Yining Zhang, Qing Lyu, Yizhong Xie, Binyang Zhang, Zhenzhong Yang, Yining Wang, et al. Gpteval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2305.14175*, 2023.

[34] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[35] Hongxia Lu, Louis Ehwerhemuepha, and Cyril Rakovski. A comparative study on deep learning models for text classification of unstructured medical notes with various levels of class imbalance. *BMC medical research methodology*, 22(1):181, 2022.

[36] Jyotsna Kumar Mandal and Debika Bhattacharya. *Emerging technology in modelling and graphics*, volume 937. Springer, 2020.

[37] Leland McInnes, John Healy, Steve Astels, et al. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205, 2017.

[38] Eneldo Loza Mencía and Johannes Furnkranz. Pairwise learning of multilabel classifications with perceptrons. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 2899–2906. IEEE, 2008.

[39] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[40] Shapol M Mohammed, Karwan Jacksi, and S Zeebaree. A state-of-the-art survey on semantic similarity for document clustering using glove and density-based algorithms. *Indonesian Journal of Electrical Engineering and Computer Science*, 22(1):552–562, 2021.

[41] NXP Semiconductors. Nxp official website, 2025. URL https://www.nxp.com/.

[42] Malte Ostendorff, Elliott Ash, Terry Ruas, Bela Gipp, Julian Moreno-Schneider, and Georg Rehm. Evaluating document representations for content-based legal literature recommendations. In *Proceedings of the eighteenth international conference on artificial intelligence and law*, pages 109–118, 2021.

[43] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[44] Alexandru Petrescu, Ciprian-Octavian Truică, and Elena-Simona Apostol. Language-based mixture of transformers for exist2024. *Working Notes of CLEF*, 2024.

[45] Katarzyna Poczeta, Mirosław Płaza, Tomasz Michno, Maria Krechowicz, and Michał Zawadzki. A multi-label text message classification method designed for applications in call/contact centre systems. *Applied Soft Computing*, 145:110562, 2023.

[46] Mikko Raatikainen, Quim Motger, Clara Marie Lüders, Xavier Franch, Lalli Myllyaho, Elina Kettunen, Jordi Marco, Juha Tiihonen, Mikko Halonen, and Tomi Männistö. Improved management of issue dependencies in issue trackers of large collaborative projects. *IEEE Transactions on Software Engineering*, 49(4):2128–2148, 2022.

[47] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[48] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 60(5):503–520, 2004.

[49] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[50] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[51] Matthias Schonlau, Julia Weiß, and Jan Marquardt. Multi-label classification of open-ended questions with bert. In *2023 Big Data Meets Survey Science (BigSurv)*, pages 1–8. IEEE, 2023.

[52] Philip Sedgwick. Spearman's rank correlation coefficient. *Bmj*, 349, 2014.

[53] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[54] Kristina P Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE access*, 8:80716–80727, 2020.

[55] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[56] Christoph Stach. Data is the new oil–sort of: a view on why this comparison is misleading and its implications for modern data administration. *Future Internet*, 15(2): 71, 2023.

[57] Adane Nega Tarekegn, Mario Giacobini, and Krzysztof Michalak. A review of methods for imbalanced multi-label classification. *Pattern Recognition*, 118:107965, 2021.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[59] Sai Pavan Kumar Veeranki, Akhila Abdulnazar, Diether Kramer, Markus Kreuzthaler, and David Benjamin Lumenta. Multi-label text classification via secondary use of large clinical real-world data sets. *Scientific Reports*, 14(1):26972, 2024.

[60] Gissel Velarde, Anindya Sudhir, Sanjay Deshmane, Anuj Deshmunkh, Khushboo Sharma, and Vaibhav Joshi. Evaluating xgboost for balanced and imbalanced data: application to fraud detection. *arXiv preprint arXiv:2303.15218*, 2023.

[61] Bailin Wang, Lin Qiu, Shomir Choudhury, H. Andrew Zhang, and Michael C. Hughes. Are large language models fair judges? *arXiv preprint arXiv:2305.19397*, 2023.

[62] Adi Widianto, Eka Pebriyanto, Fitriyanti Fitriyanti, and Marna Marna. Document similarity using term frequency-inverse document frequency representation and cosine similarity. *Journal of Dinda: Data Science, Information Technology, and Data Analytics*, 4(2):149–153, 2024.

[63] Rüdiger Wirth and Jochen Hipp. Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, volume 1, pages 29–39. Manchester, 2000.

[64] Jianxin Wu. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 5(23):495, 2017.

[65] Shuo Xu, Yuefu Zhang, Xin An, and Sainan Pi. Performance evaluation of seven multi-label classification methods on real-world patent and publication datasets. *Journal of Data and Information Science Vol*, 9(2), 2024.

[66] Hang Yan, Mingxue Ma, Ying Wu, Hongqin Fan, and Chao Dong. Overview and analysis of the text mining applications in the construction industry. *Heliyon*, 8(12), 2022.

[67] Min-Ling Zhang and Zhi-Hua Zhou. A k-nearest neighbor based algorithm for multi-label classification. In *2005 IEEE international conference on granular computing*, volume 2, pages 718–721. IEEE, 2005.

[68] Shengnan Zhang and Huiyang Xu. Multi-label text classification method based on reberta-textcnn. In *Fourth International Conference on Computer Vision, Application, and Algorithm (CVAA 2024)*, volume 13486, pages 599–604. SPIE, 2025.

[69] Xian-Da Zhang and Xian-Da Zhang. Support vector machines. *A Matrix Algebra Approach to Artificial Intelligence*, pages 617–679, 2020.

[70] Jure Zupan. Introduction to artificial neural network (ann) methods: what they are and how to use them. *Acta Chimica Slovenica*, 41(3):327, 1994.

Appendix A

Classification Results

This appendix presents the full classification results obtained during the evaluation of the different models, vectorization methods, and classification strategies. The table summarizes the performance metrics across all tested configurations, including accuracy, recall, precision, F1-score, and additional multi-label evaluation metrics. These results support the performance comparison discussed in Section 4.

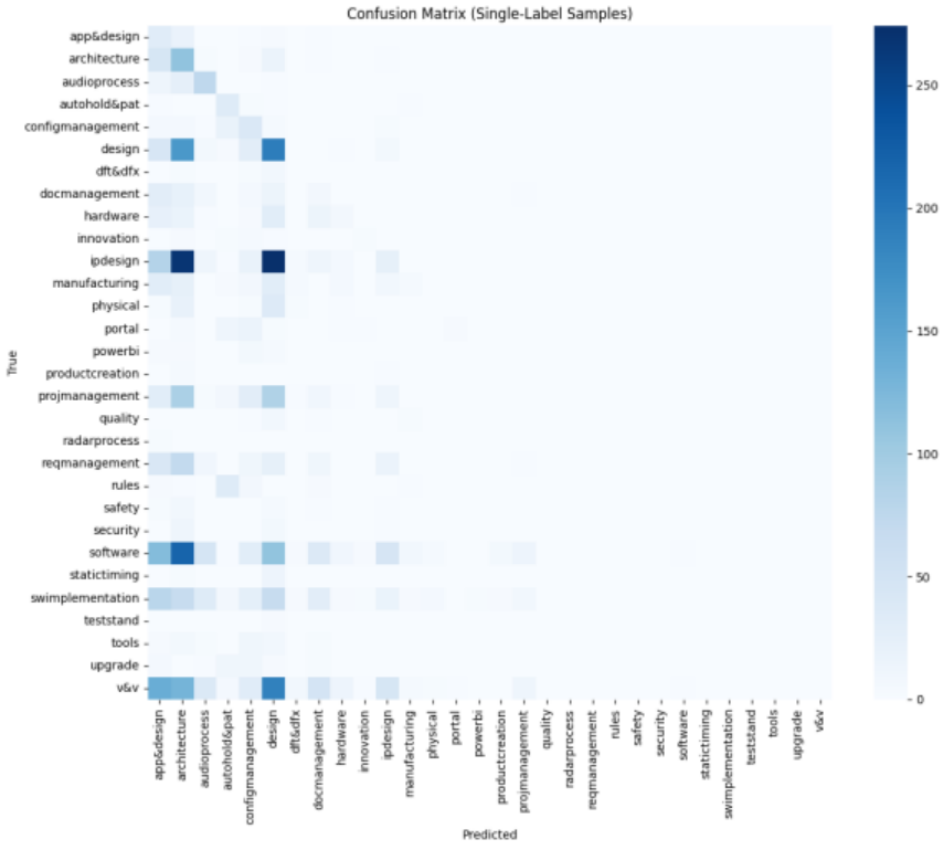


Figure A.1: Confusion matrix for single-label samples (DistilBERT).

Vectorizer	Transformation	Classifier	Single Labels Correct (4589)	Multi-labels Correct (270)	MPP	MAP	Coverage Error	Recall@3	Recall@5	Recall@7	Recall@10	F1-Score	Hamming Loss
		BERT	4240 (92.39)	212 (78.52)	0.1892	0.7621	2.2197	0.8599	0.9209	0.9580	0.9800	0.3125	0.1380
		RoBERTa	4205 (91.63)	216 (80.00)	0.1878	0.7368	2.3745	0.8347	0.9139	0.9443	0.9730	0.3103	0.1385
		DistilBERT	4260 (92.83)	221 (81.85)	0.1904	0.7601	2.2122	0.8572	0.9261	0.9556	0.9778	0.3145	0.1376
TF-IDF	OvR	Logistic Regression	3991 (86.97)	179 (66.30)	0.1772	0.6384	3.0673	0.7597	0.8647	0.9138	0.9560	0.2959	0.1420
TF-IDF	BR	Logistic Regression	4005 (87.27)	180 (66.67)	0.1778	0.6374	3.0667	0.7575	0.8677	0.9155	0.9553	0.2940	0.1418
TF-IDF	CC	Logistic Regression	3262 (71.08)	148 (54.81)	0.1449	0.5085	5.1538	0.5804	0.7068	0.7841	0.8604	0.2395	0.1528
TF-IDF	LP	Logistic Regression	3906 (85.12)	212 (78.52)	0.1740	0.6173	3.2054	0.7273	0.8500	0.9034	0.9464	0.2889	0.1428
TF-IDF	OvR	Ridge Classifier	3759 (81.91)	174(64.44)	0.1671	0.5974	4.1528	0.7041	0.8151	0.8656	0.9062	0.2762	0.1454
TF-IDF	CC	Ridge Classifier	3118 (67.95)	169 (62.59)	0.1397	0.4912	6.7451	0.5703	0.6783	0.7399	0.7923	0.2305	0.1545
TF-IDF	LP	Ridge Classifier	2369 (51.62)	135 (50.00)	0.1064	0.3816	18.9930	0.4791	0.5163	0.6001	0.6179	0.1756	0.1656
TF-IDF	OvR	Multinomial NB	3965 (86.40)	162 (60.00)	0.1754	0.6124	3.1676	0.7266	0.8576	0.9151	0.9592	0.2901	0.1426
TF-IDF	LP	Gaussian NB	1872 (40.79)	85 (31.48)	0.0832	0.2603	23.0750	0.3538	0.4059	0.5097	0.5488	0.1375	0.1734
TF-IDF	OvR	Random Forest	3989 (86.93)	174 (64.44)	0.1769	0.6542	3.3161	0.7692	0.8637	0.9147	0.9521	0.2925	0.1421
TF-IDF	LP	kNN	2497 (54.41)	112 (41.48)	0.1109	0.3679	16.7674	0.4915	0.5410	0.6364	0.6709	0.1833	0.1641
TF-IDF	CC	XGBoost	3691 (80.43)	147 (54.44)	0.1631	0.5644	3.8460	0.6864	0.7978	0.8710	0.9309	0.2698	0.1467
TF-IDF	LP	XGBoost	3786 (82.50)	163 (60.37)	0.1678	0.5923	3.6605	0.7071	0.8196	0.8781	0.9309	0.2775	0.1451
TF-IDF	OvR	MLP	3888 (84.72)	183 (67.78)	0.1730	0.6429	3.6112	0.7403	0.8435	0.8871	0.9266	0.2859	0.1434
TF-IDF	CC	MLP	3582 (78.06)	167 (61.85)	0.1593	0.6015	4.7818	0.6839	0.7767	0.8268	0.8718	0.2633	0.1480
TF-IDF	CC	LinearSVC	3567 (77.73)	163 (60.37)	0.1585	0.5858	4.4769	0.6711	0.7731	0.8360	0.8881	0.2620	0.1482
TF-IDF	OvR	SVM	3733 (81.35)	181 (67.04)	0.1663	0.6018	3.8974	0.6965	0.8104	0.8704	0.9150	0.2749	0.1456
TF-IDF (n-grams)	OvR	Logistic Regression	4012 (87.43)	172 (63.70)	0.1778	0.6458	2.9911	0.7685	0.8684	0.9293	0.9698	0.2940	0.1418
TF-IDF (n-grams)	OvR	Ridge Classifier	3785 (82.48)	178 (65.93)	0.1684	0.6196	4.1309	0.7276	0.8209	0.8680	0.9062	0.2783	0.1449
TF-IDF (n-grams)	CC	Ridge Classifier	3385 (73.76)	167 (61.85)	0.1509	0.5552	5.8230	0.6414	0.7352	0.7862	0.8316	0.2494	0.1508
TF-IDF (n-grams)	LP	Ridge Classifier	2551 (55.59)	133 (49.26)	0.1140	0.4386	17.5562	0.5231	0.5546	0.6280	0.6444	0.1883	0.1631
TF-IDF (n-grams)	OvR	Multinomial NB	3972 (86.55)	168 (62.22)	0.1739	0.6205	3.1141	0.7403	0.8597	0.9144	0.9599	0.2910	0.1424
TF-IDF (n-grams)	CC	Multinomial NB	3893 (84.83)	177 (65.56)	0.1729	0.6180	3.3038	0.7294	0.8438	0.9027	0.9450	0.2859	0.1434
TF-IDF (n-grams)	LP	Multinomial NB	3980 (86.73)	159 (58.89)	0.1759	0.6250	3.1438	0.7429	0.8605	0.9145	0.9576	0.2910	0.1425
TF-IDF (n-grams)	CC	XGBoost	3704 (80.71)	151 (55.93)	0.1638	0.5686	3.8039	0.6878	0.8011	0.8705	0.9293	0.2710	0.1465
TF-IDF (n-grams)	LP	XGBoost	3767 (82.09)	175 (64.81)	0.1675	0.5938	3.6851	0.7098	0.8167	0.8792	0.9297	0.2768	0.1452
TF-IDF (n-grams)	OvR	MLP	3862 (84.16)	182 (67.41)	0.1718	0.6426	3.6344	0.7413	0.8379	0.8878	0.9241	0.2840	0.1438
TF-IDF (n-grams)	CC	MLP	3573 (77.86)	164 (60.74)	0.1588	0.6046	4.8222	0.6842	0.7745	0.8225	0.8686	0.2625	0.1481
TF-IDF (n-grams)	OvR	SVM	3726 (81.19)	175 (65.81)	0.1658	0.6227	3.8950	0.7188	0.8083	0.8630	0.9062	0.2740	0.1458
TF-IDF (n-grams)	CC	LinearSVC	3623 (78.95)	165 (61.11)	0.1610	0.6093	4.4600	0.6934	0.7852	0.8375	0.8848	0.2661	0.1474
TF-IDF (n-grams)	LP	LinearSVC	2757 (60.08)	108 (40.00)	0.1217	0.4985	16.1850	0.5669	0.5961	0.6508	0.6674	0.2015	0.1605
Count Vectorizer	OvR	Logistic Regression	3749 (81.70)	165 (61.11)	0.1663	0.5848	3.6172	0.6847	0.8121	0.8851	0.9396	0.2750	0.1456
Count Vectorizer	OvR	Ridge Classifier	2865 (62.43)	127 (47.04)	0.1271	0.4407	6.8150	0.5136	0.6211	0.7003	0.7791	0.2103	0.1587
Count Vectorizer	OvR	Multinomial NB	4014 (87.47)	166 (61.48)	0.1777	0.6370	2.9807	0.7613	0.8637	0.9230	0.9616	0.2939	0.1418
Count Vectorizer	CC	XGBoost	3665 (79.86)	146 (54.07)	0.1619	0.5697	3.8506	0.6863	0.7921	0.8664	0.9306	0.2679	0.1471
Count Vectorizer	OvR	LinearSVC	3610 (78.67)	166 (61.48)	0.1604	0.5578	4.0665	0.6582	0.7825	0.8556	0.9138	0.2652	0.1476
Sentence-BERT	OvR	Logistic Regression	3795 (82.70)	171 (63.33)	0.1685	0.5888	3.4657	0.6922	0.8224	0.8910	0.9452	0.2786	0.1449
Sentence-BERT	CC	Logistic Regression	3552 (77.40)	153 (56.67)	0.1574	0.5548	4.0374	0.6402	0.7691	0.8453	0.9157	0.2604	0.1486
Sentence-BERT	OvR	Ridge Classifier	3791 (82.61)	169 (62.59)	0.1683	0.5831	3.6779	0.6951	0.8214	0.8799	0.9327	0.2782	0.1450
Sentence-BERT	CC	Random Forest	3780 (82.37)	147 (54.44)	0.1669	0.5902	3.9620	0.7057	0.8166	0.8811	0.9295	0.2761	0.1455
Sentence-BERT	LP	Random Forest	3624 (78.97)	139 (51.48)	0.1599	0.5511	4.5254	0.6631	0.7829	0.8535	0.9062	0.2646	0.1478
Sentence-BERT	LP	kNN	2875 (62.65)	137 (50.74)	0.1280	0.4434	13.9592	0.5863	0.6237	0.6924	0.7177	0.2115	0.1584
Sentence-BERT	CC	XGBoost	3907 (85.14)	152 (56.30)	0.1725	0.5752	3.3461	0.7193	0.8444	0.9048	0.9520	0.2853	0.1436
Sentence-BERT	LP	XGBoost	3804 (82.89)	157 (58.15)	0.1683	0.5655	3.6433	0.6879	0.8229	0.8908	0.9372	0.2784	0.1450
Sentence-BERT	CC	MLP	3744 (81.59)	195 (72.22)	0.1674	0.6567	3.6153	0.7207	0.8138	0.8703	0.9225	0.2764	0.1453
Sentence-BERT	OvR	LinearSVC	3761 (81.96)	166 (61.48)	0.1669	0.5726	3.5237	0.6792	0.8147	0.8906	0.9474	0.2759	0.1455
Word2Vec	OvR	Logistic Regression	3220 (70.17)	146 (54.07)	0.1430	0.4768	4.9382	0.5546	0.6979	0.7937	0.8784	0.2364	0.1534
Word2Vec	OvR	Ridge Classifier	3349 (72.98)	141 (52.22)	0.1483	0.4699	4.8317	0.5657	0.7249	0.8083	0.8881	0.2453	0.1616
Word2Vec	LP	Gaussian NB	2496 (54.39)	150 (55.56)	0.1124	0.3645	7.6027	0.4118	0.5449	0.6399	0.7442	0.1855	0.1636

Table A.1: Multi-label Classification All Result

Vectorizer	Transformation	Classifier	Single Labels Correct (4589)	Multi-labels Correct (270)	MPP	MAP	Coverage Error	Recall@3	Recall@5	Recall@7	Recall@10	F1-Score	Hamming Loss
Word2Vec	CC	Random Forest	3414 (74.40)	129 (47.78)	0.1505	0.5047	5.1100	0.6115	0.7373	0.8225	0.8884	0.2491	0.1509
Word2Vec	LP	Random Forest	3180 (69.30)	123 (45.56)	0.1403	0.4475	5.9004	0.5568	0.6869	0.7732	0.8553	0.2322	0.1543
Word2Vec	LP	kNN	2306 (50.25)	120 (44.44)	0.1031	0.3085	18.2477	0.4389	0.5014	0.5989	0.6409	0.1702	0.1667
Word2Vec	CC	XGBoost	3541 (77.16)	128 (47.41)	0.1559	0.4978	4.1861	0.6178	0.7638	0.8552	0.9247	0.2580	0.1491
Word2Vec	LP	XGBoost	3491 (76.07)	136 (50.37)	0.1541	0.4935	4.4056	0.6062	0.7543	0.8414	0.9098	0.2550	0.1497
Word2Vec	CC	MLP	3232 (70.43)	173 (64.07)	0.1447	0.5318	5.0710	0.5918	0.7030	0.7766	0.8532	0.2389	0.1529
Word2Vec	LP	MLP	3698 (80.58)	178 (65.93)	0.1647	0.5520	3.9369	0.6684	0.8023	0.8691	0.9211	0.2721	0.1462
Word2Vec	OvR	LinearSVC	3265 (71.15)	138 (51.11)	0.1446	0.4624	4.7200	0.5413	0.7066	0.8105	0.9043	0.2391	0.1529
GloVe	OvR	Logistic Regression	3035 (66.14)	124 (45.93)	0.1342	0.4380	5.4024	0.5092	0.6565	0.7617	0.8605	0.2221	0.1563
GloVe	OvR	Ridge Classifier	3094 (67.42)	131 (48.52)	0.1370	0.4310	5.3399	0.5214	0.6697	0.7787	0.8651	0.2267	0.1554
GloVe	LP	Gaussian NB	2260 (49.25)	127 (47.04)	0.1014	0.3307	9.0958	0.3756	0.4927	0.5775	0.6754	0.1675	0.1673
GloVe	CC	Random Forest	3241 (70.63)	115 (42.59)	0.1426	0.4607	5.8413	0.5568	0.6992	0.7817	0.8596	0.2361	0.1535
GloVe	LP	Random Forest	2997 (65.31)	110 (40.74)	0.1320	0.4223	6.6365	0.5169	0.6469	0.7344	0.8179	0.2185	0.1571
GloVe	LP	kNN	2029 (44.21)	95 (35.19)	0.0902	0.2606	20.0716	0.3676	0.4401	0.5454	0.6013	0.1492	0.1710
GloVe	CC	XGBoost	3387 (73.81)	118 (43.70)	0.1489	0.4654	4.6703	0.5776	0.7305	0.8262	0.9040	0.2466	0.1514
GloVe	LP	XGBoost	3372 (73.48)	135 (50.00)	0.1490	0.4724	4.7300	0.5689	0.7290	0.8208	0.8964	0.2465	0.1514
GloVe	CC	MLP	3122 (68.03)	147 (54.44)	0.1389	0.5001	5.5904	0.5568	0.6768	0.7590	0.8349	0.2295	0.1548
GloVe	LP	MLP	3534 (77.01)	167 (61.85)	0.1573	0.5146	4.4481	0.6311	0.7666	0.8430	0.8999	0.2599	0.1487
GloVe	OvR	LinearSVC	3084 (67.20)	123 (45.56)	0.1363	0.4183	5.2484	0.4957	0.6669	0.7730	0.8802	0.2255	0.1557
FastText	OvR	Logistic Regression	3070 (66.90)	138 (51.11)	0.1363	0.4486	5.303	0.5208	0.6653	0.7603	0.8592	0.2254	0.1556
FastText	OvR	Ridge Classifier	3417 (74.46)	153 (56.67)	0.1517	0.4792	4.5791	0.5904	0.7404	0.8201	0.8979	0.2508	0.1505
FastText	LP	Gaussian NB	2187 (47.66)	135 (50.00)	0.0987	0.3148	9.6431	0.3533	0.4776	0.5685	0.6594	0.1627	0.1682
FastText	CC	Random Forest	3444 (75.05)	121 (44.81)	0.1515	0.5112	5.0612	0.6177	0.7428	0.8256	0.8900	0.2508	0.1506
FastText	LP	Random Forest	3247 (70.76)	106 (39.26)	0.1425	0.4689	5.7446	0.5734	0.6998	0.7805	0.8552	0.2360	0.1536
FastText	LP	kNN	2048 (44.63)	98 (36.30)	0.0912	0.2699	19.5163	0.3861	0.4440	0.5529	0.5934	0.1507	0.1707
FastText	CC	XGBoost	3614 (78.75)	132 (48.89)	0.1592	0.5115	4.0954	0.6341	0.7798	0.8623	0.9242	0.2634	0.1480
FastText	LP	XGBoost	3581 (78.03)	133 (49.26)	0.1578	0.5142	4.2569	0.6254	0.7731	0.8517	0.9171	0.2612	0.1485
FastText	CC	MLP	3276 (71.39)	165 (61.11)	0.1462	0.5433	5.0276	0.6042	0.7116	0.7818	0.8586	0.2415	0.1523
FastText	LP	MLP	3778 (82.33)	168 (62.22)	0.1677	0.5646	3.7516	0.6915	0.8185	0.8818	0.9309	0.2772	0.1452
FastText	OvR	Linear SVC	3241 (70.63)	150 (55.56)	0.1441	0.4635	4.7308	0.5420	0.7028	0.8012	0.9010	0.2382	0.1530
Word2Vec + FastText	OvR	Logistic Regression	3596 (78.36)	150 (55.56)	0.1592	0.5069	3.9938	0.6186	0.7781	0.8662	0.9330	0.2633	0.1480
Word2Vec + FastText	OvR	Ridge Classifier	3646 (79.45)	155 (57.41)	0.1615	0.5320	4.0142	0.6364	0.7894	0.8651	0.9237	0.2672	0.1472
Word2Vec + FastText	OvR	Multinomial NB	3040 (66.25)	140 (51.85)	0.1351	0.4184	5.6554	0.5065	0.6592	0.7473	0.8407	0.2234	0.1560
Word2Vec + FastText	LP	Gaussian NB	3200 (69.73)	190 (70.37)	0.1440	0.4534	4.9486	0.5457	0.6978	0.7907	0.8753	0.2376	0.1531
Word2Vec + FastText	CC	Random Forest	3818 (83.20)	160 (59.26)	0.1690	0.5906	3.9649	0.7162	0.8263	0.8910	0.9380	0.2796	0.1447
Word2Vec + FastText	LP	XGBoost	3744 (81.59)	157 (58.15)	0.1658	0.5559	3.7661	0.6730	0.8103	0.8822	0.9380	0.2742	0.1458
Word2Vec + FastText	LP	MLP	3886 (84.68)	179 (66.30)	0.1727	0.5828	3.4262	0.7105	0.8425	0.9016	0.9434	0.2855	0.1435
Word2Vec + FastText	OvR	LinearSVC	3639 (79.30)	156 (57.78)	0.1612	0.5360	3.7856	0.6512	0.7877	0.8792	0.9394	0.2667	0.1473
Word2Vec + FastText		ML-kNN	3499 (76.25)	156 (57.78)	0.1553	0.5207	4.6966	0.6208	0.7578	0.8416	0.8862	0.2566	0.1493
Word2Vec + FastText		RankSVM	3657 (79.69)	139 (51.48)	0.1613	0.5219	3.9191	0.6434	0.7927	0.8769	0.9378	0.2679	0.1471
Word2Vec + FastText		MMP	3850 (83.90)	152 (56.30)	0.1700	0.5597	3.464	0.6781	0.8304	0.8960	0.9536	0.2809	0.1445
Sentence-BERT		ML-kNN	3733 (81.35)	169 (62.59)	0.1658	0.5986	3.9405	0.6822	0.8088	0.8784	0.9114	0.2740	0.1458
Sentence-BERT		RankSVM	3461 (75.42)	119 (44.07)	0.1521	0.4801	4.3750	0.5876	0.7462	0.8430	0.9201	0.2519	0.1504
Sentence-BERT		MMP	3913 (85.27)	168 (62.22)	0.1734	0.5981	3.3325	0.7267	0.8474	0.9044	0.9482	0.2868	0.1433
TF-IDF		RankSVM	3370 (73.44)	101 (37.41)	0.1475	0.5121	4.4725	0.6019	0.7249	0.8235	0.9135	0.2443	0.1519
TF-IDF		MMP	3909 (85.18)	186 (68.89)	0.1747	0.6330	3.3200	0.7452	0.8470	0.9000	0.9419	0.2871	0.1432
		ANN	3053 (66.53)	100 (37.04)	0.1340	0.4438	5.8211	0.5482	0.6575	0.7379	0.8320	0.2218	0.1564
		RNN	2886 (62.89)	99 (36.67)	0.1268	0.3509	6.4408	0.4453	0.6220	0.7217	0.8107	0.2100	0.1588
		CNN	3799 (82.78)	141 (52.22)	0.1674	0.6191	3.6170	0.7282	0.8203	0.8729	0.9224	0.2771	0.1453
		CNN-RNN	3771 (82.17)	119 (44.07)	0.1653	0.6094	3.7642	0.7142	0.8119	0.8723	0.9224	0.2738	0.1460
		BiLSTM	3809 (83.00)	120 (44.44)	0.1669	0.6104	3.7313	0.7243	0.8202	0.8697	0.9155	0.2766	0.1454
		CNN-BiLSTM	3736 (81.41)	112 (41.48)	0.1635	0.5998	3.9401	0.7059	0.8040	0.8575	0.9081	0.2709	0.1466

Appendix B

Association Results

	Not enough info.	Not useful	Partially useful	Useful
BM25				
Prediction 1	3.12%	17.19%	14.84%	64.84%
Prediction 2	3.91%	36.72%	19.53%	39.84%
Prediction 3	4.30%	49.22%	15.23%	31.25%
Prediction 4	4.69%	57.03%	14.45%	23.83%
Prediction 5	4.35%	56.92%	18.18%	20.55%
Cosine Similarity				
Prediction 1	4.38%	21.91%	13.55%	60.16%
Prediction 2	3.19%	38.65%	17.53%	40.64%
Prediction 3	3.59%	45.42%	21.51%	29.48%
Prediction 4	4.38%	53.39%	20.72%	21.51%
Prediction 5	4.38%	56.97%	19.92%	18.73%
BERTopic				
Prediction 1	3.19%	21.12%	13.55%	62.15%
Prediction 2	2.79%	43.82%	17.93%	35.46%
Prediction 3	5.18%	49.80%	19.52%	25.50%
Prediction 4	2.79%	60.96%	16.73%	19.52%
Prediction 5	4.78%	63.35%	15.94%	15.94%

Table B.1: Rating Distributions per Rank per Model