Exploring Design Discussions With Semi-Supervised Topic Modelling

by

Roshan Neil Lasrado
B.E., University of Mumbai, 2016

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Exploring Design Discussions With Semi-Supervised Topic Modelling

by

Roshan Neil Lasrado
B.E., University of Mumbai, 2016

Supervisory Committee

Dr. Neil A. Ernst, Supervisor
(Department of Computer Science)

Dr. Daniela Damian, Departmental Member
(Department of Computer Science)

ABSTRACT

Stack Overflow is a rich source of questions and answers—discussions—about software development. One topic of discussion is software design, such as the correct use of design patterns or best practices in data access. Since design is a more abstract topic in software engineering, researchers have long sought to characterize and model design knowledge. However, these approaches typically require significant expert input to contextualize the abstract design information. In this study, we explore how combining expert input with Stack Overflow might serve as an effective way to identify design topics. Being able to identify and classify this design knowledge would enable the discovery and sharing of this knowledge, enabling developers better leverage Stack Overflow for crowd-sourcing their design decisions. We first perform inductive coding of design-tagged Stack Overflow questions and answers to identify the design concepts that developers discuss. We report on areas where inter-rater agreement was a challenge, including abstraction levels. Since inductive coding is expensive, we apply a semi-supervised (Anchored CorEx) approach. We find that it outperforms LDA and offers superior interpretability and the ability to incorporate expert domain knowledge. We leverage Anchored CorEx to identify how design is discussed on Stack Overflow and leveraged in GitHub projects. We conclude by describing how our experience using the semi-supervised CorEx approach leads us to believe that approaches like Anchored CorEx that combine domain knowledge and scalability are key for analyzing large SE text repositories.

Table of Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to:

Dr. Neil A. Ernst, my supervisor, for his invaluable guidance, patience, kindness, and counsel, that helped me think critically about my work. His immense knowledge and expertise, so generously shared, helped me grow as a researcher. I am extremely grateful for this amazing opportunity and also for his words of support and encouragement especially during trying times. I also thank him for the resources and funding provided to support my work.

Dr. Daniela Damian, and the **defence committee members** for their interest in my work.

Dr. Margaret-Anne Storey, for the initial feedback and guidance with respect to this work, and for the remarkable *CSC 578A Empirical Software Engineering* course for giving me a head start in research methods.

Rohith Pudari, Ze Shi (Zane) Li, Dave Cheng, and my fellow colleagues at the Octera Research Group, for their support with this research and for the fun, collaborative environment, and engaging discussions that kept me inspired and motivated to do some good work. I am grateful to **Alvi Mahadi** and **Karan Tongay** for their support and mentoring during the initial phases of my study.

University of Victoria and the **Department of Computer Science** for providing a vibrant environment to study and conduct research. I would like to acknowledge the smooth transition online and the support provided during the pandemic. I am extremely grateful for the rewarding graduate school experience.

DEDICATION

Theresa and Ronny, my dear mom and dad,
Rohan, dear brother, to you too,
Anita, for encouraging and inspiring this lad,
With love, I dedicate this work to you.

Chapter 1

Introduction

Software design is a crucial component of the software development activity since it determines the various aspects of the system such as its performance, maintainability, robustness, security, etc. A poor design could lead to technical debts incurring penalties ranging from additional developer efforts, capital, resources, and security incidents, to loss of market share and a negative public opinion. Representing software design as a collection of design decisions and the rationale for those decisions would not only help enable sharing, compliance, discovery and traceability of the design choices [17], but would also make deliberations over the choices easier.

Furthermore, software design is a socio-technical process involving various stakeholders reasoning and making decisions [17]. This highlights the collaborative nature of software design that involves deliberations rather than being based only on experience. These software design discussions form an important part of the iterative software development process. Several studies [5, 15, 67, 52, 79, 78, 83, 48] have analyzed design discussions as occurring on various developer communication channels such as issues, pull requests, code reviews, mailing lists, and chat messages, for various purposes such as identifying latent design decisions, design challenges, design quality etc.

Stack Overflow[1] is the most popular question and answer (Q&A) forum used by developers for their software development queries [7]. This large footfall for Stack Overflow can be attributed to it being easily accessible (e.g. Stack Overflow links are top Google search results for software development related queries) and widely referenced (e.g. IDEs like Google Colab having a *Search Stack Overflow* option in

[1] `https://stackoverflow.com/`

response to errors). Stack Overflow hosts a wide range of software development related posts and provides software developers with all levels of experience, a highly democratized platform for conducting discussions and deliberations in the form of questions, answers, and comments. These features make Stack Overflow a prime candidate for software developers to crowd-source their design decisions / discussions.

Identifying design-related discussions and categorizing them would not only help make the wealth of information available on platforms like Stack Overflow, more accessible but would enable the development of certain practical solutions such as simplifying the crowd-sourcing of software design, automated documentation of design decisions, etc. But classifying design discussions is a challenging task. In the next section, we discuss this challenge and the research opportunity that it creates as the motivation for this study.

1.1 Motivation

Concerns about software design remain difficult to understand. This is because software design is one of the least concrete elements in the software development lifecycle, at least compared to testing, implementation, and deployment [10]. Software design is usually done heuristically by pulling from the current project context (e.g., the architecturally significant requirements), the design team's knowledge, and an ever-changing set of patterns and styles from the literature. How this process works, what makes one design good and another bad, and how to support developers in design tasks is subjective and frequently changes as new technologies emerge. To examine how software developers discuss design topics, we conducted an exploratory study using Stack Overflow.

The emergence of Stack Overflow as a knowledge repository for many software-related topics, such as energy management [59], suggests that software design knowledge also occurs there. Indeed, Soliman et al. [70] show that this is true for the specific case of architecture knowledge for middleware. We expand on their study for the more general case of software design knowledge and derive various software design-related areas that the developers find challenging. For this, we extend the Stack Overflow design-related dataset, created by a previous study [45], using a *Tag Filtering* approach. Using the question tags of Stack Overflow allows us to leverage a low-cost annotation directly created by the developers themselves [72].

The biggest challenge to characterizing latent design topics on Stack Overflow is

that software design discussions, decisions, and knowledge encompass a broad scope of topics, including sub-areas such as architecture choices, design patterns, and data access, to name a few. Design knowledge can be specific to a particular API or language (e.g., JavaScript's Promises framework) or generic for many possible implementations (such as Publish-Subscribe). Recovering design knowledge involves finding design information that is often scattered and poorly organized in software projects [17].

Another advantage of categorizing design knowledge in Stack Overflow, compared to project-specific communication platforms, such as pull requests or issue comments, is access to a larger community of developers and easy access to design discussions not limited by the project's context. Developers could potentially leverage this immense wealth of knowledge for design documentation, which is a challenging task. Design documentation in projects is often missing or incomplete [20]. If a developer chooses a novel or complicated design approach (e.g. use of dependency injection from the C# framework), ideally he or she would comment on this in the code or add an Architecture Decision Record (ADR, [55]) to the repository explaining their choice. However, these notes get outdated quickly and are not updated [53]. Documentation requires time better spent writing code and thus is rarely done. Leveraging on-demand documentation principles [64], by contrast, would potentially allow this burden to be lessened. One avenue for dynamically generating documentation might be to extract the essence of a specific design choice from a large body of design knowledge. In this thesis, we investigate the various ways in which the Stack Overflow design related posts are referenced in GitHub projects. If developers are already using the Stack Overflow references for documentation, then tools and processes could be enhanced to make this process even more effective.

1.2 Problem Statement and Research Questions

The overarching goal of this study is to:

> Identify the various types of software design related topics discussed on Stack Overflow. Toward this goal, we explore how combining expert input for classifying Stack Overflow posts could serve as an effective way to identify design topics. Additionally, we seek to establish the usefulness of this design knowledge by demonstrating ways in which it can be leveraged.

As discussed in the previous sections, software design is notoriously difficult to classify into design topics, due to design being subjective and composed of several overlapping topics. The objective of this study is to achieve a better classification of design knowledge in Stack Overflow, to enable researchers and practitioners to use this knowledge. In addition, this knowledge can be classified using various manual and automated approaches. Our objective here is to additionally identify the approach that would provide a better classification of the design topics. We explore whether using a semi-supervised approach (Anchored CorEx [77, 62, 26]) would result in performance improvements over the most commonly used topic modelling technique to classify Stack Overflow posts (LDA [16]) [68]. We conduct an exploratory study with the following research objectives:

RQ1: What are effective techniques to classify design related posts on Stack Overflow?

Approach— We categorize design-related posts using four approaches. For **RQ1.1**, we examine posts using a manual inductive coding approach. We explore how accurate this approach is, and find that it can be very challenging to get an agreement. We conduct an additional investigation to determine the causes of the low agreement and report our findings.

In **RQ1.2** we use an unsupervised approach, LDA topic modelling [16] and an anchored, semi-supervised approach called Anchored CorEx [77, 62, 26]. We compare these two approaches in their precision, recall, F1 score, and balanced accuracy, and contrast them with the inductive labels we created earlier and the user tags from Stack Overflow. In addition, we present a discussion on the evaluation metrics to be used in this scenario and an argument for the necessity of a balanced approach.

RQ2: How can we leverage knowledge of design topics discussed on Stack Overflow?

Approach— While there are many applications of the semi-supervised Anchored CorEx approach, we look at two practical implications of the results of RQ1: we apply the semi-supervised design classification approach arrived at in RQ1 on the entire Stack Overflow dataset. We discuss the topic frequency and co-occurrence. Further, since developers can refer to Stack Overflow in software artifacts such as code comments, we investigate to what extent such links are

to design discussions as identified above, characterize the type of discussion referred to, and document the ways in which GitHub projects use design-related Stack Overflow references.

1.3 Research Design and Methodology

For this study, we use our tag-filtering approach (see Section 3.2.1) to identify design-related posts. In addition, we use a coding guide, arrived at after multiple rounds of inductive coding, to define software design and its various types. We base our definition of what constitutes design inductively using the key topics identified during inductive coding based on our experiences as professional software developers, the inter-coder agreement, and our literature review of existing studies performing design mining [45, 23, 71, 78].

Further, we choose a Stack Overflow post (question along with its associated answers) as the level of granularity for our analysis. We make this choice to have better visibility of the design discussion between the asker of the question and the users responding with an answer. Since a Stack Overflow post is focused in terms of the topic of discussion and is marked by tags to highlight the area of discussion (and for visibility to users answering the question), we believe this to be a good choice for the level of granularity for our analysis.

We begin by assessing the **best way to extract and characterize design-related knowledge** in Stack Overflow. We first create a dataset of design-related posts. We then compare and contrast four methods of classifying Stack Overflow posts: (1) qualitative, manual, inductive coding; (2) using existing Stack Overflow user tags; and (3) unsupervised and (4) semi-supervised topic modelling. Each of these four approaches can potentially classify Stack Overflow posts into design related topics. We explore the differences in the design topics that each method identifies. We show how using a semi-supervised approach can bring the best of both highly scalable LDA approaches and domain knowledge of inductive coding.

We then report on what **design topics are being discussed on Stack Overflow**. We propose some ways in which knowing these topics can be used in a practical way. One is to use these topics to identify what (design-related) questions are being linked in GitHub projects and report the various ways in which the projects are utilizing the posts for documentation purposes. If developers are already using Stack Overflow links in repositories for documentation, then this implies that tools and

processes could be enhanced to make this process even more effective.

We conclude by providing guidelines for future studies utilizing semi-supervised topic modelling and Stack Overflow as a source of design discussions, and discussing the limitations of our work.

1.4 Contributions

This thesis contributes the following:

- We demonstrate a semi-supervised learning approach to extract design topics from Stack Overflow. We show that Anchored CorEx outperforms LDA in terms of recall and balanced accuracy. We compare the various topic modelling strategies used and present our discussion on the evaluation metric most suitable for future studies using a similar context like ours.
- We explore practical applications of the resulting topic model, including a characterization of the design discussions on Stack Overflow, including an analysis of co-occurring topics, and the various ways in which GitHub projects document and use Stack Overflow design references.
- We release our tagged dataset from Stack Overflow consisting of over 200,000 posts, and our inductively coded set of 150 posts used to seed Anchored CorEx. We make this available at 10.5281/zenodo.5885783.
- Based on our experiences in this study, we present guidelines for future studies that use semi-supervised topic modelling in SE. Additionally, we also discuss the advantages and challenges of sourcing design knowledge from Stack Overflow posts.

1.5 Thesis Outline

This thesis is organized as follows:

Chapter 2 elaborates the background information and some related work on mining Stack Overflow and managing software design knowledge. It further introduces the topic modelling strategies that would be explored in this thesis.

Chapter 3 introduces our comprehensive Stack Overflow design discussions dataset. It then presents the results of (1) Inductive Coding, (2) Unsupervised (LDA), and (3) Semi-Supervised (Anchored CorEx) approaches in classifying Stack Overflow software design discussions.

Chapter 4 presents the results of applying the technique identified as most suitable for mining software design (Anchored CorEx) to two scenarios: (1) to gain insights into Stack Overflow design discussions and (2) to learn how design discussions are leveraged by developers in their projects.

Chapter 5 compares the various design topic classification approaches used and presents a discussion regarding the evaluation metrics. In addition, we provide guidelines for future studies using semi-supervised topic modelling in SE and for using Stack Overflow as a repository of design knowledge. We conclude by discussing the practical and research implications of this study and by identifying the limitations.

Chapter 6 presents the conclusion of this research study.

Chapter 2

Background and Related Work

2.1 Introduction

In this chapter, we first highlight some related work on mining Stack Overflow and related sites. Further, we discuss existing research on extracting and managing software design knowledge. Finally, we conclude by looking at the topic modelling strategies commonly used in software engineering research and provide some details regarding the topic modelling strategies that we further explore in this thesis.

2.2 Mining Stack Overflow

Stack Overflow is the most popular question and answer (Q&A) forum online [7]. A study by Mamykina et al. [46] attributes this success of Stack Overflow to factors such as the highly responsive and iterative design of the platform and its high visibility. Software developers too often rely on crowd-sourced Q&A sites like Stack Overflow for completing software development tasks [39]. Due to this popularity of Stack Overflow, it has been an active subject of many research studies targeted at its improvement (such as improvements to comment presentation [84], support for improving the chance of the question being answered [51], etc.). Vasilescu et al. [76] present a substantial collection of research involving Stack Overflow. All these factors hint at the relevance of Stack Overflow for Software Engineering and its role in affecting the design of the systems that developers build.

To study the role of Stack Overflow in software development, several studies have looked at how GitHub artifacts and Stack Overflow content are linked. Yang et

al. [81] investigated how Stack Overflow code snippets (specifically Python) appeared in GitHub code repositories in a large-scale study. They found evidence that 1-2% of the code from Stack Overflow appears on GitHub, suggesting that the transfer occurs via copy and paste programming; this implies that a similar mechanism may occur for the design concepts we detect (but which would be more challenging to uniquely identify in code). In contrast, specific links from code to Stack Overflow are easy to identify and are explicitly provided in the SOTorrent dataset. Manes and Baysal [47] show this occurs quite often, with a median of 144 references per project. Software design was not one of the more popular topics they identified, but like us they found developers seem to point more often to the question as a way of reference for code changes. Abdalkareem et al. [2], in their study, envision Stack Overflow to be increasingly used by developers as a crowd-sourcing platform to refine their software design. Another study by Liargkovas et al. [40], studying the educational gap in software engineering between academia and industry, found that developers often refer to Stack Overflow for knowledge units related to software design.

Stack Overflow is the largest software development Q&A forum, but has limitations: it can only be used by a subset of practitioners, is English and omits questions that are proprietary or internal (e.g., referring to internal APIs). Like user-generated content, it is subject to the fallibility of its users, including off-topic comments, cliques, and bias toward majority groups [24]. Other developer forums such as Gitter [22] and GitHub Discussions [31], may be relevant in future design mining studies.

2.3 Finding and Managing Design Knowledge

In the context of software engineering, design is traditionally understood both as a process [25] in which a development team engages, and as the resulting specifications [61] that the team produces. Over the natural evolution of a software system, small changes accumulate, causing the actual design of the system to differ from what was originally documented, a process known as *design erosion* [75]. At the same time, the software developer's overall knowledge of a system's design is often subject to "evaporation," which causes the developers to gradually lose knowledge of the design over time [63].

The idea that useful design information may occur in discussions is supported by the literature on design rationale [44]. Initial research on design and its rationale focused on how to create a useful representation, generally involving either an observer

being present at the moment the design was discussed or an act of reconstruction executed after the design phase had been completed [37].

Organizing design information was a primary motivation behind the patterns community. Until quite recently, this knowledge was organized largely manually, both because the information was heavily context-specific, but also due to a lack of large datasets. More recently, Gorton et al. semi-automatically populated design knowledge from internet sources for a particular (big data) domain [30], Tian et al. [74] used Stack Overflow to find how architecture smells were discussed, Bi et al. [12] looked at architecture patterns mentioned on Stack Overflow, and Ali et al. [4] uses Stack Overflow to classify posts according to architectural lifecycle stage (analysis, synthesis, evaluation, etc.). We focus primarily on design topics and their relationship to source code artifacts.

This thesis primarily studies design discussions that occur on public Q&A forum, Stack Overflow, that software developers frequently visit during their software development activity [39]. Other artifacts besides question and answer posts are also relevant. Viviani et al. [78] attempted to recover design information from the social artifacts surrounding software engineering. They focused on developer discussions that take place on GitHub pull requests, as concerns related to design are often raised in those discussions.

Most closely related to our work is that of Mohamed Soliman and his co-authors. They have looked at using Stack Overflow as an architecture knowledge (AK) repository: first, in [70], assessing the "suitability of capturing and reusing of AK for technology decisions from Stack Overflow posts", specifically middleware knowledge. Their expert respondent pool agreed that their sample of Stack Overflow posts was architecturally relevant. Next, they build an ontology for architecture knowledge on Stack Overflow [69]. This ontology is similar to the codes that we identified during inductive coding, but includes more architecturally specific terms such as *component, connector, architecture pattern* and then applies that ontology to a query-based retrieval system [71]. This last paper showed that questions with specific design details were harder to retrieve.

The primary difference with our thesis is that we look more broadly at all design topics and use a semi-supervised approach to guide the topic model. We also use the term 'design', as opposed to their focus on architectural terms. We suspect design-related discussions may be more frequent, but these two terms do have considerable overlap. On Stack Overflow, there are currently 16,325 questions tagged

'[architecture]' (as of 30-03-2022), out of the 1.5 million or so we consider in our dataset (which includes the '[architecture]' tag).

2.4 Topic Modelling in Software Engineering Research

Topic modelling is an unsupervised (and *semi-supervised*) approach to recovering latent topics in the text. It has the benefit of being highly scalable (e.g., can handle all of Stack Overflow) and has a long history in software engineering research studies [68, 3, 73, 18].

Topic modelling has been widely used in software engineering for analyzing textual data as a part of empirical software engineering studies, as well as for improving software development activities [68]. Among these, a large number of studies have used Stack Overflow as a data source. Stack Overflow has also been a part of various mining challenges (including the MSR 2015 challenge), with researchers mining it to study subjects ranging from software security [82], machine learning [8], energy consumption [59], to non-functional requirements [85], among others. These studies adopt a similar approach to ours in using text analysis tools such as LDA [14, 16] to analyze the dataset, once it is filtered using keyword labels.

In this thesis, we compare the conventional unsupervised approach, LDA, to a newer semi-supervised approach, CorEx, which uses anchoring. Weak supervision (or semi-supervised) refers to the fact that users do some labelling, but much less than inductive coding.

2.4.1 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) was introduced as a topic modelling technique by Blei et al. [14] and is commonly used as an unsupervised technique for identifying latent topics in text corpora, using the statistical properties of word frequencies and co-occurrences [41]. In a recent study by Silva et al. (2021) [68] LDA and LDA-based techniques were found to be the most commonly used topic modelling technique in software engineering research.

As shown in Figure 2.1, LDA topic modelling [14, 16] assumes D documents contain T topics expressed with W different words. Each document $d \in D$ of length N_d is modeled as a discrete distribution $\theta(d)$ over the set of topics. Each topic

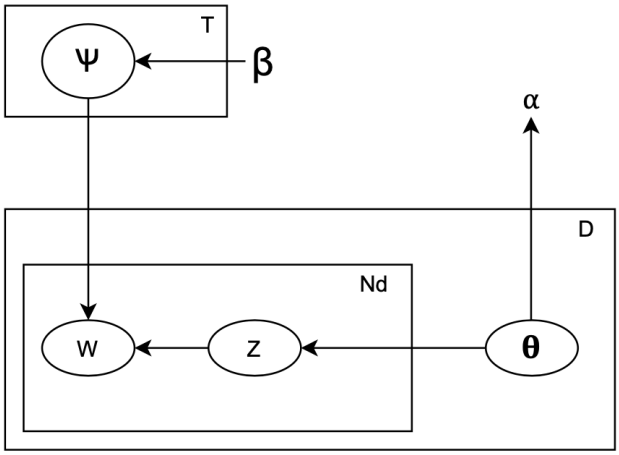

Figure 2.1: LDA topic modelling architecture.

corresponds to a multinomial distribution over the words. α is the discrete prior assigned to the distribution of topics vectors(θ); and β for the distributions of words in topics(ψ). The inference problem in LDA is to find hidden topic variables z, a vector spanning all instances of all words in the dataset.

Although LDA appears to be a promising topic modelling approach, studies such as the one by Agarwal et al. [3] caution regarding the instability and limited validity of LDA topics. We explore how LDA performs for modelling software design-related topics.

2.4.2 Anchored Correlation Explanation (Anchored CorEx)

Anchored Correlation Explanation (Anchored CorEx) [77, 62, 26] is a semi-supervised approach for topic modelling, that allows incorporating domain knowledge in topic modelling by specifying anchor words. Introduced by Gallagher et al. (2017) [26], it provides an alternative to LDA's generative assumptions by using an information-theoretic approach to learning topics.

Anchored CorEx involves optimizing the equation:

$$Maximize \quad TC(X;Y) + \beta \sum_{i,j \in R} I(X_i, Y_j) \tag{2.1}$$

where $TC(X;Y)$ represents the total correlation with the objective of constructing latent variables Y that best explain the multivariate dependencies in the data X, and

$I(x, y)$ represents the maximizing of mutual information between the anchor terms X_i and the latent factors Y_j.

The Anchored CorEx topic modelling, as opposed to LDA, is discriminative in nature and estimates the probability of a document belonging to a topic, given that document's words, by using a binary softmax. Whereas LDA, as a generative model, models the actual distribution of the dataset. Therefore, for Anchored CorEx, a document to be classified is evaluated individually against each of the topics as a binary classification problem. This means that a given document could potentially belong to all of the topics, none of the topics or a combination of topics. We hypothesize that this nature of Anchored CorEx would make it more suitable for modelling the highly overlapping nature of software design.

Anchored CorEx has been used as a topic modelling technique in a variety of domains (Human factors in Aviation Reports [43], Business Trends from academic papers [58], Cryptocurrency fraud analysis [54], Covid-19 perception and impact-related studies [38, 49, 66, 19], etc.). As a part of Software Engineering research, a recent study by Cogo et al. used Anchored CorEx for analyzing the gap between a developer's information needs and the programming language documentation [21]. To the best of our knowledge, ours is the first study that uses Anchored CorEx for the domain of software design. We apply this newer topic modelling approach for mining software design topics and in doing so share our experiences and guidelines for future software engineering research.

2.5 Chapter Summary

In this chapter, we first provided some background on Stack Overflow, the Q&A forum that we would use as the basis of our study in this thesis. We discussed some of the existing studies using Stack Overflow as a data source for research, and quoting some of the previous work, established Stack Overflow as the most popular discussion/reference site used by software developers for completing their software development tasks [68], thus being a potential source for software design discussions. To further explore the role of Stack Overflow in software development, we reviewed studies that have investigated its role in open source projects on GitHub. We learnt about the prevalence of Stack Overflow references in GitHub projects and about the design-related posts on Stack Overflow. We then discussed some limitations of Stack Overflow.

Furthermore, we reviewed some of the previous work in the domain of software design. We established the importance of finding and managing design knowledge by discussing *design erosion* [75] and *evaporation* [63]. We then reviewed an existing study by Soliman et al. [70, 69, 71] that is closely related to our study. In our thesis, we look more broadly at all design topics and explore an unsupervised (LDA) and a semi-supervised approach (Anchored CorEx) for topic modelling.

Finally, we reviewed the use of topic modelling strategies in software engineering research. We learnt of the wide-scale use of topic modelling (particularly of LDA and LDA-based methods [68]) for a variety of software engineering aspects. We concluded by discussing the two topic modelling approaches that we explore in this thesis (LDA and Anchored CorEx) and by discussing some of the other studies that use these topic modelling techniques.

In the next chapters, we apply the topic modelling techniques discussed here to Stack Overflow discussions to identify the best algorithmic approach to identifying software design-related topics discussed (Chapter 3). We then apply the identified technique to classify the design discussions on Stack Overflow and learn how these discussions are leveraged by developers in their projects (Chapter 4). Finally, we discuss some of the practical implications of our findings (Chapter 5).

Chapter 3

Design Mining using Topic Modelling

3.1 Introduction

In this chapter, we begin by explaining our approach to **RQ-1** (What are effective techniques to classify design-related posts on Stack Overflow?). To achieve this, we categorize design-related posts using three different approaches and evaluate the results and advantages of the approaches under study. In the following chapters, we compare the three approaches for categorizing design-related posts with a fourth approach of using user-defined Stack Overflow tags and discuss some of its challenges.

We introduce a new comprehensive Stack Overflow design discussions dataset and elaborate on our tag-filtering approach used to create the dataset (Section 3.2.1). We then explain our methodology for the three software design topic classification approaches under study: Inductive Coding (Section 3.2.2), Topic Modelling using Latent Dirichlet Allocation (LDA) (Section 3.2.3.1), and Topic Modelling using Anchored Correlation Explanation (Anchored CorEx) (Section 3.2.3.2).

We observe that manually classifying Stack Overflow software design-related posts using inductive coding is challenging and prone to disagreements among coders. We conduct additional analysis to identify the cause of disagreements and in doing so shed some light on the nature of software design discussions on Stack Overflow (Section 3.3.1.1). We then compare the results of the two topic modelling approaches (LDA and Anchored CorEx) using precision, recall, F1 score and balanced accuracy values with our inductively coded labels and the Stack Overflow user-defined tags

(Section 3.4). We find that the discriminative semi-supervised Anchored CorEx topic modelling approach outperforms and provides some advantages over the generative unsupervised LDA topic modelling approach.

3.2 Methodology

In this section, we explain our methodology to address **RQ-1** (What are effective techniques for classifying design-related posts on Stack Overflow?), including dataset creation using tag filtering, inductive coding, and topic modelling using LDA and Anchored CorEx. All of the following analysis and evaluation was carried out using the Google Colab[1] platform with a custom backend hosted on Google Cloud (e2-standard-16: 16 vCPU, 64 GB memory, 200 GB persistent disk) and Google BigQuery for data hosting, querying, and analysis.

3.2.1 Dataset Creation

Identifying software design-related discussions on Stack Overflow is a challenging task and is heavily dependent on the user-defined tags for filtering. A review of existing design mining studies by Mahadi et al. [45] revealed that on average 14% of any software corpus (such as comments on GitHub pull requests, commits & issues, IRC logs, code comments, etc.) studied consisted of design discussions. A similar trend was also observed for a previous Stack Overflow design-related dataset (Mahadi et al. [45]) created using 10 software design-related tags, where the prevalence of software design discussions was 15%.

Stack Overflow follows a community-driven question and answer framework, wherein details such as appropriate creation and use of tags are monitored by expert users. Additionally, the number of tags that can be assigned to a topic is limited to 5. These rules ensure that the tags that are assigned to a post remain germane to the topic of discussion. We, therefore, propose utilizing a tag filtering approach (as described below) for creating a comprehensive Stack Overflow design-related dataset.

The Stack Overflow dataset was obtained using the latest data dump (as of writing this thesis) of the *SOTorrent dataset* (version: 2020-12-08) [7] and queried using the Google BigQuery platform. Refer [7] for more background on mining Stack Overflow.

[1] `https://colab.research.google.com/`

3.2.1.1 Tag filtering

To create a comprehensive Stack Overflow design discussions dataset, we use a two-step approach. We extend the Stack Overflow design tag set created by a previous study [45] by performing related tag analysis to identify tags and subtags that are related to the already identified design tags. Further, to identify software design areas and tags that were missed out by related tag analysis, we query the Stack Overflow tag description (as identified by PostTypes: `TagWiki` and `TagWikiExcerpt`) for the keyword "design" and having more than 100 posts. We review the resultant set of 1694 Stack Overflow user-defined tags, considering their tag descriptions and a random set of posts under each tag, to gain a better understanding of tags and to exclude the tags that are not related to software design. Additionally, we avoid including domain/technology-specific tags such as [wordpress] and [php], since such tags contained both design-related and non-design-related posts. Also, tags corresponding to graphic design and UI design (e.g., [css], [html]) are also excluded.

The systematic identification of design tags produced a set of 61 Stack Overflow tags corresponding to software design. These tags are then used to identify design-related questions and their corresponding answers that form a part of our comprehensive Stack Overflow design discussions dataset. Table 3.1 lists the identified Stack Overflow tags. The extended design discussions dataset contains 626,701 design-related questions and answers.

3.2.1.2 Data Processing

Preprocessing is performed on the posts to remove styling tags, code blocks and whitespace, to make the dataset easier for a human labeller to perform manual analysis and labelling. Only question and answer type posts are retained in the dataset, along with their Ids, titles and associated tags. Table 3.2 presents a sample Stack Overflow design tagged question type post after preprocessing.

3.2.2 Inductive Coding

We begin by using a manual inductive coding approach [50] for identifying the distribution of the design topics in our dataset (**RQ1.1**). The initial intention of this step was to obtain a set of ground-truth labels for other automatic classification approaches. However, as discussed in Section 3.3.1, it was observed that manually

abstract-factory	dependency-inversion	liskov-substitution-principle	separation-of-concerns
adapter	design-patterns	locking	single-responsibility-principle
api-design	design-principles	mediator	singleton
architectural-patterns	domain-driven-design	memento	software-design
architecture	double-checked-locking	model-view-controller	solid-principles
bridge	facade	module	state-pattern
builder	factory	multiton	strategy-pattern
builder-pattern	factory-method	null-object-pattern	system-design
chain-of-responsibility	flyweight-pattern	observer-pattern	template-method-pattern
change-data-capture	front-controller	oop	threadpool
class-design	interface-segregation-principle	open-closed-principle	unit-of-work
code-design	interpreter-pattern	parameter-object	visitor
command-pattern	inversion-of-control	prototype-pattern	visitor-pattern
composite	iterator	proxy-pattern	
decorator	language-design	reactor	
dependency-injection	lazy-initialization	scheduler	

Table 3.1: The list of 61 Stack Overflow design-related tags identified using the two-step approach and used to create the comprehensive Stack Overflow design-related dataset.

Table 3.2: Sample Stack Overflow design tagged post after initial preprocessing.

Post Id	Title	Body
43474516	How stores objects update view in flux pattern	Imagine we have a chat application and conversation page has been opened. If one of messages edited by other user or message's state changed from sent to deliver, Action update Store with new messages metadata. For example,after these actions we have a list of messagesState or messagesText or simply messages with modified data in our Store . So in this scenario we don't know which row has been edited and we render all the data in view again. Is this behavior one of Flux principles? Isn't better to update and send event about updated object only? (I developing Android application and so I don't use reactJS or other library like this) Also I going to think it's good if we mix MVP with Flux! because if one view want to change itself we have to put logic in view.for example view directly get store data and check it belongs to which element! I think a presentation layout is good for this type of situation. Has anyone tried this?

identifying design topics is a challenging task. Therefore, the results of this phase were instead used as weak-supervision for the semi-supervised approach as described in Section 3.2.3.2.

3.2.2.1 Sampling Approach

The theoretical population that this thesis is concerned with are the design topics discussed (i.e., in a question or answer) by software developers on public Q&A forums (identified using tagsets on Stack Overflow). From this purposively chosen theoretical population as the sampling frame, we randomly sample 50 questions for each round (3 total) of inductive coding to identify the design-related topics.

3.2.2.2 Methodology

We begin inductive coding [50] without having a predefined coding guide, to avoid biasing towards any specific design categories. Two graduate students each with several years of commercial software development experience independently label the posts, looking at the title, question, and answers, alongside the post tags. In each round, after reviewing 50 randomly sampled discussions independently, the coders meet in a coding agreement session to discuss codes and look for themes. A third coder (academician, expert in software design research) resolved disagreements. A coding dictionary is maintained and updated after each round of coding for the set of codes identified. Table 3.4 shows our coding dictionary and relative code frequency out of the 150 questions coded for Stack Overflow.

Since no new labels were identified after three rounds of coding, reconciliation, and agreement (50 posts per round; 150 posts in total), the set of labels was finalized and the 150 posts were re-labelled using these finalized labels. Table 3.3 reports the agreement kappa scores for the Design Topic labels assigned. Section 3.3.1 discusses more on agreement.

3.2.3 Topic Modelling

In the previous section, we discussed a manual approach to classifying design discussions. In addition to the reasons cited in Section 3.3.1.1, this manual approach to classifying design is challenging and has drawbacks such as being expensive and hard to scale (requiring human efforts), and being prone to disagreements. Therefore, an unsupervised approach that requires little to no human intervention and relies solely

Table 3.3: Inter-rater agreement [36] on Design Topic identification from codes in Table 3.4.

Round	Agreement Scores (Cohen's Kappa)
Round 1	0.22 (fair)
Round 2	0.17 (slight)
Round 3	0.34 (fair)

on statistical textual models would be more suitable for design classification. In this section, we compare the conventional unsupervised approach, LDA, with a newer semi-supervised approach, CorEx, which uses anchoring (**RQ1.2**).

3.2.3.1 Latent Dirichlet Allocation (LDA)

The LDA topic modelling [16] was carried out using an existing implementation called Mallet[2] to automatically identify topics of design-related posts on Stack Overflow.

Here, we first experimentally preprocess our dataset to improve LDA performance and to ensure that the model picks relevant words for topic representation. Our dataset consists of 227,282 design questions (along with their associated answers) with a vocabulary size of 259,197. We perform stop word removal, lemmatization, and bigram and trigram tokenization.

Lemmatization is the process of normalizing a given word to its base/root form (e.g. lemmatized form of 'walking' is 'walk'). Whereas *Stemming* simply removes the last few characters of the word, resulting in a word that may have incorrect spelling or may not be the correct morphological root of the original word (e.g. stemming the word 'caring' may result in 'car'). For our analysis, we choose to perform lemmatization over stemming because various studies (e.g. [35, 6]) have demonstrated with experiments that lemmatization outperforms stemming in terms of precision for information retrieval tasks and clustering.

This reduces our vocabulary size to 213,329 words. To further narrow down the vocabulary to improve the model performance, we add the high-frequency words in the dataset to the stop words list and filter out words that occurred in fewer than 10 documents, or more than 50% of the documents. This further reduced the vocabulary to 30,457 words.

[2] `http://mallet.cs.umass.edu`

Table 3.4: Inductive, manually assigned design-related codes. Counts are number of questions (out of 150) associated with that code.

Codes	Description	SO
Other Design Questions	Discussion around efficiency, security and various best/recommended practice design choices.	31
OOP Design	Discussion around designing of classes and objects along with their associated attributes and methods.	19
Dependency Management	The various challenges of management and insertion of dependencies.	18
Architecture	Better ground up design of multiple systems/modules and their interaction/integration.	10
Design Patterns	General reusable solution to some commonly occurring software design problem e.g. GoF patterns [27].	10
Language Design	Challenges of language design and the discussion around programming constructs, code refactoring or code smells.	8
Data Storage Access Design	Design choices around the selection of appropriate data structure/iteration strategies for storage/representation/accessing data.	7
Test Design	Design of test cases and their challenges.	4
Design Principles	Software design principles that mainly focus on the maintainability aspects of quality.	4
API Design	Design choices and challenges involved in developing APIs .	2
Not Design	*The post did not pertain to design despite its tag.*	37

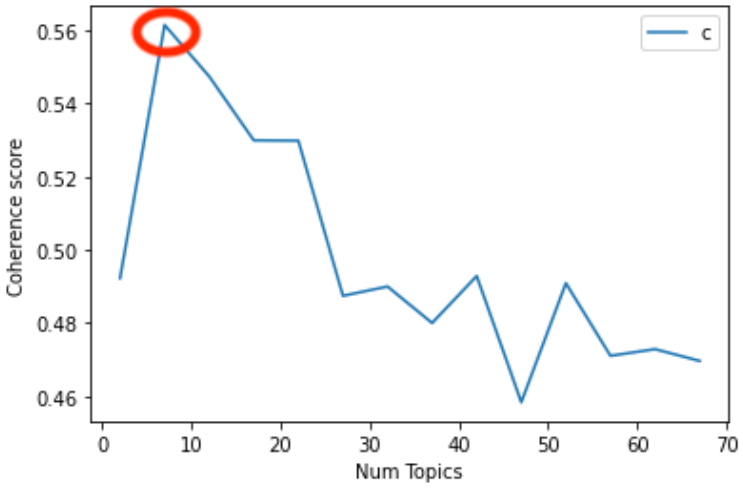

Figure 3.1: Coherence scores for LDA topics. The circled value was chosen as the optimal topic size (7).

We used Mallet's optimization technique to determine the hyperparameter for the optimal number of topics for the given design discussions dataset by comparing their C_v coherence scores. We perform a grid search for the number of topics between the limits that we define as max (70) and min (2) topic numbers to retrieve the number of topics with the highest coherence score. These seem a reasonable prior to apply based on studies in SE using LDA (such as [8]). Figure 3.1 shows the coherence graphs we used to determine the optimal topic counts, with the chosen values circled. Seven topics are optimal for the Stack Overflow dataset, with a coherence score of 0.56.

3.2.3.2 c-TF-IDF and Anchored Correlation Explanation (Anchored CorEx)

In this section, we explore a semi-supervised method for topic modelling, by using a combination of Class-Based Term Frequency-Inverse Document Frequency (c-TF-IDF) and Anchored Correlation Explanation (Anchored CorEx) [77, 62, 26]. Here, the goal is to use domain knowledge (as weak supervision) with Anchored CorEx for topic modelling. We use our inductive coding results from the previous section (Section 3.3.1) (Table 3.4) as topic labels. We utilize c-TF-IDF to identify some useful anchor terms from our inductively coded data that are used as seed words for Anchored CorEx.

1. **Class Based Term Frequency-Inverse Document Frequency (c-TF-IDF)**

   We use c-TF-IDF to identify the top 30 informative terms (i.e., candidate anchor terms for CorEx, below) for each of the classes identified in the inductively coded dataset. The terms identified for each of the classes provide a latent representation of the classes as terms that can strongly distinguish a class. c-TF-IDF is a TF-IDF-like method for feature representation, where each document represents a class. The c-TF-IDF for every term t and a class document d is given as:

   $$c - tf - idf(t, d) = tf(t, d) * (log\frac{1 + N}{1 + df(t)} + 1) \tag{3.1}$$

   where $tf(t, d) = f_{t,d}/\sum_{f' \in d} f_{t',d}$ is the term frequency of the term t in the class document d with $f_{t,d}$ term count, N is the number of class documents, and $df(t)$ is the document frequency of the term t (the number of class documents containing the term t).

   We preprocess the inductively coded Stack Overflow dataset by performing lemmatization, unigram and bigram tokenization, and generate the class documents by merging posts based on their assigned codes. The standard scikit-learn[3] TfidfVectorizer is then used on the class documents to generate the c-TF-IDF values which are then ranked to extract the top 30 informative terms per class.

   The top 30 informative terms are then manually analyzed and filtered to retain only meaningful terms (for that label) and to eliminate spurious correlations. Table 3.5 presents the inductive labels, showing the top 30 terms identified per class and the meaningful terms retained to be used as anchor words for Anchored CorEx (in blue). For example, in Topic 8 (Not-Design), words such as 'entity' or 'module' are clearly design-related, and thus we do not choose them as meaningful representations for the Not-Design topic. On the other hand, words like 'songwriter' reflect the sparsity of the dataset and, while not design, nonetheless seem unlikely to anchor topics properly (in our design exploration, anyway).

[3] `https://scikit-learn.org`

Table 3.5: Top 30 representative terms for a selection of inductively defined design topics (Table 3.4), retrieved using c-TF-IDF. Blue are the selected anchor terms. Black terms are discarded as anchors. Red are anchor terms added to overcome topic overlap.

Topic Id	Topic	Terms
1	api-design	layer, keras, success, status, weight, status code, tensorflow, server, access layer, angular, able access, api, code, error, callback call, angular resource, iterate layer, dataset, tensor, call, model, define class, define, access, use, able, example, network, resource, method, rest, restful, soap
2	architecture	datablock, content, dll, class, assembly, abstraction, use, version, web, application, delegate, app, interface, block, model, datum, service, object, need, assembly version, concern, not, menu, file, wpf, app delegate, main app, method, system, architecture
3	data-storage-access-design	author, table, model, book, authorbook, datum, author book, code, relationship, use, cshtml, database, question, db, normaltime, illustration, customerid, getstatus, need, time, file, cakephp, join, metadata, edge, would, query, loop, create, set
4	dependency-mgmt	composition root, service, composition, dependency, use, inject, unity, net, root, di, injection, container, asp, asp net, dependency injection, viewmodel, object, create, need, datum, controller, instance, constructor, implementation, php, not, class, interface, error, resolve
5	design-pattern	node, command, view, class, pattern, object, render, update, method, implement, scene, not, strategy, scenegraph, scene graph, state, throw, rendergraph, opengl, issuer, interceptor, command pattern, pipeline, model, type, graph, singleton, event, submit, transition

Topic Id	Topic	Terms
6	design-principles	deck, card, design, structured, structured design, focus, design principle, card deck, workspace, card collection, middle, model, principle, user, object, parent instance, middle entity, deck model, collection, author, software design, instance, reference, bind, scope, hierarchy, parent, module, not, new, principle
7	language-design	iterator, enumeration, iterable, class, collection, iterator object, generator, minion, list, method, not, don, java, use, object, dictionary, remove, new, call, function, need, return, name, decorator, give iterator, collection enumeration, iterator extend, extend enumeration, want, pattern
8	not-design	not, class, method, use, error, try, server, file, module, verilog, model, event, php, entity, songwriter, image, code, work, window, table, js, need, variable, constructor, self, database, framework, text, want, self songwriter, syntax, validation, exception
9	oop	class, method, not, object, constructor, employee, vector, argument, need, static, superclass, function, base class, atm, subclass, instance, static method, inherit, base, use, mutable, design, inheritance, create, value, input, php, object orient, call, code
10	other-design-questions	aggregate, use, not, domain, need, object, person, state, model, element, lock, class, way, email, value, event, file, saga, domain model, service, example, entity, create, time, user, address, change, want, component, executor, aggregate root, thread, lock, threadpool, concurrency, iterator, iterable
11	test-design	test, table, lock, database, bean, thread, drop, override, helper, service, table exist, exist, diagnose, drop table, write helper, typical solution, helper table, ngresource, enter, unit test, solution write, typical, angularjs, config, statement, primary, resource, xml, check, not, test configuration, test config, diagnose

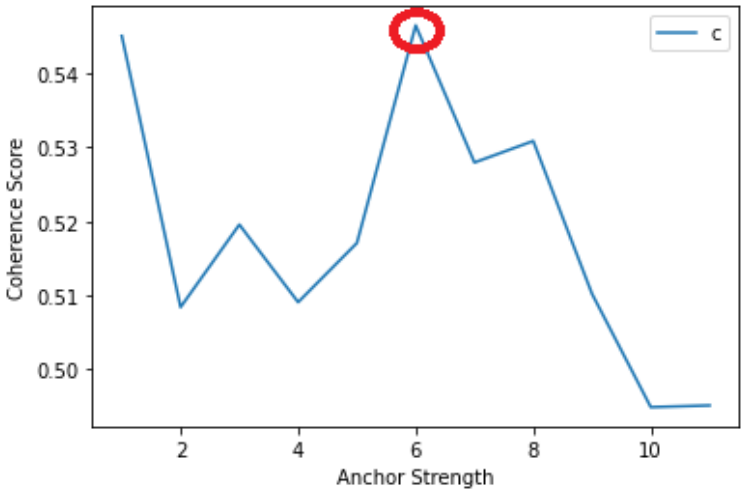

Figure 3.2: Coherence scores for CorEx anchor strength. The circled value was chosen as the anchor strength (6).

2. **Anchored Correlation Explanation (Anchored CorEx)**
   We use the terms identified through c-TF-IDF as weak supervision anchor words that seed and impose semantics on latent factors while performing Anchored CorEx [77, 62, 26].

   For Anchored CorEx topic modelling, we preprocess the dataset, performing stop-word removal, lemmatization, and unigram, bigram tokenization. We then use the Anchored CorEx implementation[4] and perform simple grid-search hyperparameter optimization for coherence scores (C_v) by adjusting for anchor strength, a measure of how strong the anchor term is.

   In case of overlapping topics (due to a spuriously correlated term), we add the overlapped term to the appropriate topic as a new anchor. Figure 3.2 shows that an anchor strength of 6 provides the highest coherence score of 0.55 and a total correlation score of 60.11.

3.3 Results

In this section, we first discuss the results of Inductive coding and our analysis of why it is difficult to manually classify software design topics. We then discuss the results of the two topic modelling approaches.

[4] `https://github.com/gregversteeg/corex_topic`

3.3.1 Inductive Coding

Software Design is notoriously context-specific and somewhat ephemeral. We, therefore, anticipated challenges getting agreement on labels for the design-related discussions on Stack Overflow. Table 3.4 lists our final set of labels, their short definitions, and their frequency of occurrence during inductive coding. As presented in Table 3.3, we struggled to get good inter-rater agreement, as measured by the kappa score, on the codes we assigned.

In general, the inductive coders were able to differentiate **design** and **non-design** posts well with a substantial agreement (Cohen's Kappa: 0.68) [36] prior to the alignment step in each round. These results suggest that given a Stack Overflow post, the coders found it relatively easy to identify whether it corresponded to Software Design.

Coder alignment for Design Topics (i.e., finding a more refined subcategory of Software Design) proved to be more challenging. While the coders agreed on whether a discussion artifact corresponded to software design, they were only able to reach a fair agreement before alignment, for the Stack Overflow posts, despite their background as professional software developers with experience in software design.

Even after three rounds of inductive coding and alignment, the software design labels identified by the coders overlapped and did not provide good separation in terms of inter-rater agreement. Next, we try to identify and provide an explanation for these disagreements.

3.3.1.1 Challenges with Manually Labelling Design Topics

To identify the reasons for low agreements and to shed some light on the nature of software design topics, we performed an additional round of coding using 75 randomly sampled posts, with the same three coders (that were involved in the previous rounds of coding) labelling 50 posts each with a 50% overlap with each of the other coders. In this round of coding, the coders were instructed to provide their top 2 labels for each of the posts, for analyzing topic overlap. The coding dictionary that was arrived at after the previous three rounds of coding, was used for this round. The coders were also asked to note their reasons for labelling posts and their observations if any. At the end of this round, the three coders met in a discussion to analyze the reasons for disagreement.

We expand on the identified challenges/reasons for disagreement with the per-

centage of all disagreements where this reason was cited (% Disagreements). Disagreement occurs when neither of the two labels assigned by the two coders matches. Each post may have multiple reasons for disagreement. We discuss more on the ways to overcome these challenges in Section 5.3.

Type 1: Overlapping topics (37.5 % of disagreements)
The coding dictionary had multiple possible answers for a given question. For instance, *decorator pattern* is a Gang of Four design pattern [27] but could be used as a solution for ensuring the *open-closed principle*, this represents a Design Pattern—Design Principle overlap. Another observation here is that when considering only the first (most appropriate according to the coder) of the two labels, the percentage of all posts that were disagreed upon was 49.3%. This percentage was reduced to 32% when considering the top 2 labels, suggesting an overlap of the topics.

Type 2: Questions asked were too basic (29.2%)
Some posts caused a disagreement when the question asked was judged to be too simple or naïve. This also reflects a difference in adjudging the usefulness of a question being tagged as design among the coders. For instance, a Stack Overflow question[5] discussing the basics of designing a utility class was labelled by the two coders as being *OOP Design* and *Not Design* respectively.

Type 3: Referred to different sections of the post (29.2%)
The coders cited different parts of the post to justify the different labels that were assigned. Sometimes, the person posting the question may be unaware of the underlying design issue and may incorrectly tag a particular post, or the given design problem may have multiple design solutions that could be categorized differently. It is complex to summarize a complex question with one or two topics.

Type 4: Judged error handling/bugs differently (20.8%)
Some questions seemed more like specific bug issues than design questions. For instance, a Stack Overflow question[6] discussing handling errors with respect to project dependencies was labelled by the two coders as being *Dependency Management* and *Not Design* respectively. The reason cited by the coder that

[5] `https://stackoverflow.com/questions/30019005/`
[6] `https://stackoverflow.com/questions/23663997/`

labelled it as a design-related post was that bugs like these may reflect the underlying design flaw in the system.

Type 5: Question too focused on implementation (8.3%)
Some questions seemed to be too focused on implementation and not design. For instance, a Stack Overflow question[7] asking the correct implementation of a database query was considered by one of the coders as being too specific to be considered as software design, while the other coder labeled it *Data Storage Access Design* since it discussed the efficiency and performance of the implementation. Note that here the Stack Overflow user *did* consider it design.

Type 6: Confusion due to the scale of the design problem (4.2%)
The level of abstraction of the software design question, for example, might be either Low-Level Design or High-Level Architecture. For instance, a post discussing some aspect of object-oriented design at the project level may be labelled by the coders as both *OOP Design* and *Architecture*.

3.3.2 Topic Modelling

Topic modelling, as a classification approach, is faster and vastly more scalable than inductive (human) labelling. We use the coherence score C_v as a metric for evaluating the topic models since it is strongly correlated with human labelling and is reliable for topic coherence evaluation [65].

3.3.2.1 LDA

Table 3.6 shows the results of LDA topic modelling with our manually assigned topic names. Although LDA provides an acceptable coherence score (0.56) for topic modelling [65], it does so by optimizing for coherence based on the word vector representations of the data.

The LDA analysis results in only seven design topics, which seems low given the breadth and scope of software design topics. The number of topics is considered too few as they appeared to be either too coarse-grained as compared to the other approaches, merged common topics, or did not provide good segregation of the documents in the area of interest.

[7] `https://stackoverflow.com/questions/7711432/`

Table 3.6: Stack Overflow LDA topic names (manually assigned) with top 5 terms

Topic Name	Top 5 Terms
oop	object, property, variable, instance, field
concurrency pattern	thread, call, task, request, resource
class/interface implementation issues	class, method, function, type, interface
error resolutions	module, file, dependency, work, component
other design	return, work, element, item, iterator
design patterns and principles	pattern, thing, person, language, feature
databases and architecture	datum, service, user, model, event

3.3.2.2 Anchored CorEx

Table 3.7 shows the results of Anchored CorEx topic modelling and the top 5 terms identified per topic. We do not report the anchor terms (i.e., the actual word list would be Table 3.7 + the terms in blue in Table 3.5). We note that for some of our topics, such as api-design and language design, our labelling dataset had relatively few training examples (e.g., only two questions for api-design – cf. Table 3.4). This explains some of the strange terms (like 'good' and 'think').

Anchoring results in CorEx topics that are interpretable because they can be tied back to the domain knowledge captured in the inductive coding while having coherence scores for the Stack Overflow dataset that are on par with LDA topic modelling (CorEx: 0.55, LDA: 0.56).

3.4 Discussion

In the previous section, we have discussed the results of the various topic modelling approaches under analysis. We now compare the result of the various approaches and discuss which of the two topic modelling approaches is better, i.e., which one better captures latent design discussions.

One way to answer this question is to see how well each named topic captures *existing* annotations. In Table 3.8, we use precision, recall, F1 score, and Balanced Accuracy (AUC) metrics to evaluate the performance of the two topic modelling

Table 3.7: Stack Overflow CorEx topic names and top 5 associated unigrams/bigrams, exclusive of anchors.

Topic Name	Associated Terms
api-design	good, thing, think, case, need
architecture	web, web application, framework, project, net
data-storage-access-design	view, data, controller, entity, sql
dependency-mgmt	service, container, ioc, inject dependency, resolve
design-pattern	singleton class, singleton pattern, factory pattern, pattern use, factory
design-principles	design pattern, software design, responsibility, good design, drive
language-design	create object, object not, object class, object create, orient
not-design	throw, get error, error message, throw exception, try
oop	static method, object orient, constructor, interface, class method
other-design-questions	ddd, wait, thread safe, domain object, locking
test-design	unit, mock, testing, test code, test class

approaches against the user-created tags on each question from Stack Overflow. The parentheses list the mapping between the Stack Overflow tag and the topic modelling name. Here, a 'true positive' is when a named topic from either LDA (column 1 of Table 3.6) or CorEx (column 1 of Table 3.7) matches a user-assigned tag on Stack Overflow. In some cases, we mapped a tag from Stack Overflow users (such as "solid-principles") to our named topics in LDA and CorEx (such as "design-principles" (CorEx) or "design patterns and principles" (LDA)).

Note here that the topics generated by LDA and CorEx, and the Stack Overflow tags, do not have an exact overlap in terms of the topic labels used. Table 3.8 checks the effectiveness of the approach by comparing tags that were found to be closely related. For instance, the LDA "design-patterns-and-principles" tag was observed to be closely related to the Stack Overflow user-defined tag "design-patterns".

Table 3.8: Topic Modelling precision (P), recall (R), F1 score (F1), and Balanced Accuracy (AUC) metrics. Parentheses indicate the LDA/CorEx topic name we matched on.

SO Tag	LDA (lda-topic-name)		Anchored CorEx (corex-topic-name)	
design-patterns	P: 0.25	R: 0.17	**P: 0.40**	**R: 0.74**
	F1: 0.21	AUC: 0.55	**F1: 0.52**	**AUC: 0.79**
	(design patterns and principles)		(design-pattern)	
solid-principles	P: 0.013	R: 0.27	P: 0.017	**R: 0.84**
	F1: 0.02	AUC: 0.59	F1: 0.03	**AUC: 0.82**
	(design patterns and principles)		(design-principles)	
oop	P: 0.39	R: 0.22	**P: 0.44**	**R: 0.71**
	F1: 0.28	AUC: 0.55	**F1: 0.55**	**AUC: 0.71**
	(oop)		(oop)	
architecture	P: 0.16	R: 0.35	P: 0.17	**R: 0.77**
	F1: 0.22	AUC: 0.61	**F1: 0.28**	**AUC: 0.74**
	(databases and architecture)		(architecture)	

The results in Table 3.8 are obtained by calculating the above metrics as a binary classification problem for each of the labels. This view of the dataset, is that of an imbalanced dataset with the label being analyzed belonging to the minority class. We choose to analyze the results in terms of balanced accuracy or area under the receiver operating characteristic curve (AUC-ROC or AUC) since it is a better predictor of the model performance for imbalanced datasets [42]. Furthermore, since we are interested in the minority class of the dataset, a higher recall value should be more preferable than precision. We discuss more on this evaluation approach in Chapter 5.

We see that Anchored CorEx generally has much higher recall (finds more true positives), but at the expense of lower precision (more false positives). We observe that while Anchored CorEx had a slightly higher precision than LDA, having a higher recall causes it to be a better approach in terms of the F1 score. Additionally, Anchored CorEx clearly outperforms LDA in terms of recall and balanced accuracy, the two metrics we argued to be the most apt for the current scenario, in all four cases.

Furthermore, the semi-supervised approach as an alternate method also avoids the challenges with the earlier approach of inductive coding by making use of the

themes/topics that emerged during inductive coding as weak supervision. In addition, the semi-supervised approach analyzes the entire dataset rather than just a sampled set of 150 questions that would address the agreement challenges.

While the Anchored CorEx topics, identified here, are not exhaustive of all the software design areas, they provide an overview of the design discussion areas on Stack Overflow in terms of interpretable categories as identified by domain experts while performing inductive coding. As discussed in the motivation for this thesis (Section 1.1), software design is one of the least concrete elements in the software development lifecycle [10]. In such a scenario, having a topic modelling strategy that incorporates expert input can be useful in identifying relevant topics. Further studies could opt to analyze software design through other perspectives such as *Non-Functional Requirements* including quality attributes, maintainability, security etc.

Therefore, to address **RQ-1** (What are effective techniques to classify design-related posts on Stack Overflow?), we conclude that among all the classification techniques under study in this thesis, **Anchored CorEx**, a semi-supervised topic modelling technique, is the most suitable and better performing approach for classifying design discussions on Stack Overflow.

3.5 Chapter Summary

In this chapter, we focused on addressing our **RQ-1** (What are effective techniques to classify design-related posts on Stack Overflow?). We first introduced a new comprehensive Stack Overflow design discussions dataset using our tag-filtering approach. We then evaluated and analyzed three different approaches for design topic classification of Stack Overflow posts. We learnt that manually classifying design discussions using inductive coding is difficult and prone to disagreements among the authors. We study this further and identify the cause of disagreements among the coders, thus shedding some light on the nature of software design discussions on Stack Overflow.

We then compared the two topic modelling approaches and found that while both LDA and Anchored CorEx have similar coherence scores, Anchored CorEx clearly outperforms LDA due to its semi-supervised nature, resulting in topics that are statistically optimized and provide good segregation in the area of interest. In the next chapter (Chapter 4), to establish the relevance of our findings, we apply our chosen approach (Anchored CorEx) for mining design knowledge on Stack Overflow and for studying how this knowledge is leveraged by developers in their projects on GitHub.

Chapter 4

Leveraging Design Topic Knowledge

4.1 Introduction

Having identified the best performing topic modelling approach for classifying design discussions in Stack Overflow in Chapter 3, in this chapter, we build on those findings. Among the various design topic classification techniques under study, we observed that Anchored CorEx, a semi-supervised topic modelling approach, has the best performance when mining design-related discussions on Stack Overflow, as discussed in Section 3.4 and illustrated in Table 3.8. We now aim to establish the relevance of these findings by answering **RQ-2** (How can we leverage the knowledge of design topics discussed on Stack Overflow?).

Towards this goal, we demonstrate two ways in which the design knowledge mined using Anchored CorEx can be leveraged. First, in Section 4.2, we gain insights into the design-related topics discussed on Stack Overflow. This analysis also provides an opportunity to investigate the relevance of the inductively coded labels (Section 4.2.2), the issues with the user-defined tags on Stack Overflow (Section 4.2.3), and to review the posts left unclassified by Anchored CorEx (Section 4.2.4).

Second, in Section 4.3, we demonstrate the usefulness of the Stack Overflow design knowledge corpus for software design and development. Having a suitably performant design topic classification technique enables us to search and identify the various design references in open-source projects. We identify the various ways in which developers crowd-source design discussions on Stack Overflow for various purposes

such as documentation, design / implementation reference, etc. to support their open-source projects on GitHub.

4.2 What Design-Related Topics are Discussed on Stack Overflow

We first discuss some of our findings observed while performing manual inductive coding. Using this manually labelled data to seed the anchor words for CorEx, we then extrapolate these findings to the entire Stack Overflow design discussions dataset. Table 4.2 presents some sample Anchored CorEx labelled posts.

4.2.1 Inductive Coding insights to design discussions on Stack Overflow

During the inductive coding of a set of randomly sampled Stack Overflow posts, it was observed that the posts were spread widely across various design topics, as presented in Table 3.4. The posts were observed to be more focused on specific questions related to oop (19/150) and dependency management (18/150). Additionally, despite being filtered for design-related tags, 24.6% of Stack Overflow posts were found to be not design-related. By contrast, examining non-tagged artifacts such as code comments or issue discussions (as in [45] or [78]) the inverse was found: 75-90% of the artifacts examined (including their dataset of curated Stack Overflow posts) were not design related. This shows the effectiveness of our tag filtering approach.

4.2.2 Co-occurrence of Anchored CorEx software design topics

Given the challenges with classifying design-related posts using inductive coding, the low agreement scores, and the advantages of using a semi-supervised model of classification, Anchored CorEx Topic Modelling was used for classifying design-related posts on Stack Overflow. Table 4.1 shows the number of questions associated with each of the Anchored CorEx topics.

Figure 4.1 presents the co-occurrence of the various Anchored CorEx topics in the Stack Overflow dataset labelling. Here, we observe that topics such as dependency-mgmt, not-design, other-design-questions and test-design have a comparatively lower

Table 4.1: Anchored CorEx topics. Counts reflect the number of Stack Overflow posts having some discussion around that label. The *Overlapping* counts are the number of posts that had the label as one of the multiple assigned labels. The *Exclusive* counts are the number of posts that had the label as the ONLY assigned label.

CorEx Label	SO frequency	
	Overlapping	Exclusive
oop	89,270	5,270
language-design	88,796	5,280
data-storage-access-design	74,371	7,831
architecture	73,494	5,282
not-design	65,579	10,182
api-design	63,972	899
design-pattern	54,403	2,038
design-principles	47,706	748
dependency-mgmt	38,045	2,456
other-design-questions	37,131	3,010
test-design	25,569	1,409
Unclassified Posts	-	20,419

Table 4.2: Anchored CorEx labelled posts with sample post content.

Post Id	CorEx Tags	Text
5055832	[architecture]	How many clients can servers handle at a time?... 10000 clients would connect using TCP and maintain that connection (assuming TCP has its own keepalive functionality) for 3 hours, will the server have to process anything besides keepalive...
36967157	[design-pattern,oop,test-design]	actual input is read and from which my own translation of the input is added and returned to my caller. (Is it called the Decorator pattern?)... And write unit tests for all the methods, inherited or not, of your custom reader.
50118368	[design-principles,oop]	Overwrite method or derive? For a given class, is it better/fancier/more in accordance with Python Zen to overwrite a method (I.e., assign another function to it) or derive from that class and overwrite it in the new class...
55080521	[language-design,oop]	Abstracting classes which implement the same method with different Argument types... What is the correct abstraction? How would you refactor this code?...
32214224	[not-design]	Showing form errors beside form fields... Still learning Phalcon through the tutorials but I have an issue with displaying form error messages beside form fields...

co-occurrence with other design labels (also observed as darker red cells for these topics in Figure 4.1), suggesting that the Stack Overflow posts tagged with these labels tend to be more focused and have very few latent terms associated with the other design topics, the hoped-for result for not-design.

On the other hand, we also observe that api-design, language-design and oop have a strong co-occurrence with other labels (also observed as blue cells for these topics in Figure 4.1). The strong co-occurrence for oop with other design topics may be explained by the fact that concepts such as *design principles* may also be co-tagged as oop because design principles such as *SOLID principles* are used to improve object-oriented programming. For instance, in the Stack Overflow post [1] (CorEx Topic: [design-principles,oop]), the discussion regarding an *OOP* concept regarding *attribute abstraction* touches on the *Liskov Substitution Principle* while suggesting a design solution to the problem being asked. We also observe a similar overlap in Stack Overflow post[2] (CorEx Topic: [design-pattern,oop]), between *OOP* and the *Abstract Factory design pattern*.

In the case of api-design, the analysis of the results presents spurious correlations between api-design and other design topics. These spurious correlations can be attributed to the sparsity of the inductively coded dataset for this topic as well as the non-specificity of this topic (encompassing *Web APIs* and *Language / Library APIs*).

The strong co-occurrence for the topic language-design (*programming constructs*) with other design topics can also be explained similarly. Addressing code smells or code refactoring could also address design challenges classified under other CorEx design topics. For instance, the Stack Overflow post [3] (CorEx Topic: [data-storage-access-design,language-design]) discussed a programming construct (*iterator*) for efficiently accessing (scraping) data from word documents.

Further, we also observe a strong co-occurrence between oop and language-design, which may be explained due to the *code-refactoring* and *OOP* overlap, as in the Stack Overflow post[4] (CorEx Topic: [language-design,oop]).

These findings suggest that api-design, language-design, and oop were not good distinguishers for classifying software design due to the significant overlap with the other topics. Further studies, may choose to avoid software design classification using these topics either because they represent a low level of software design (programming

[1] https://stackoverflow.com/questions/2023837/
[2] https://stackoverflow.com/questions/2747182/
[3] https://stackoverflow.com/questions/1784478/
[4] https://stackoverflow.com/questions/18805056/

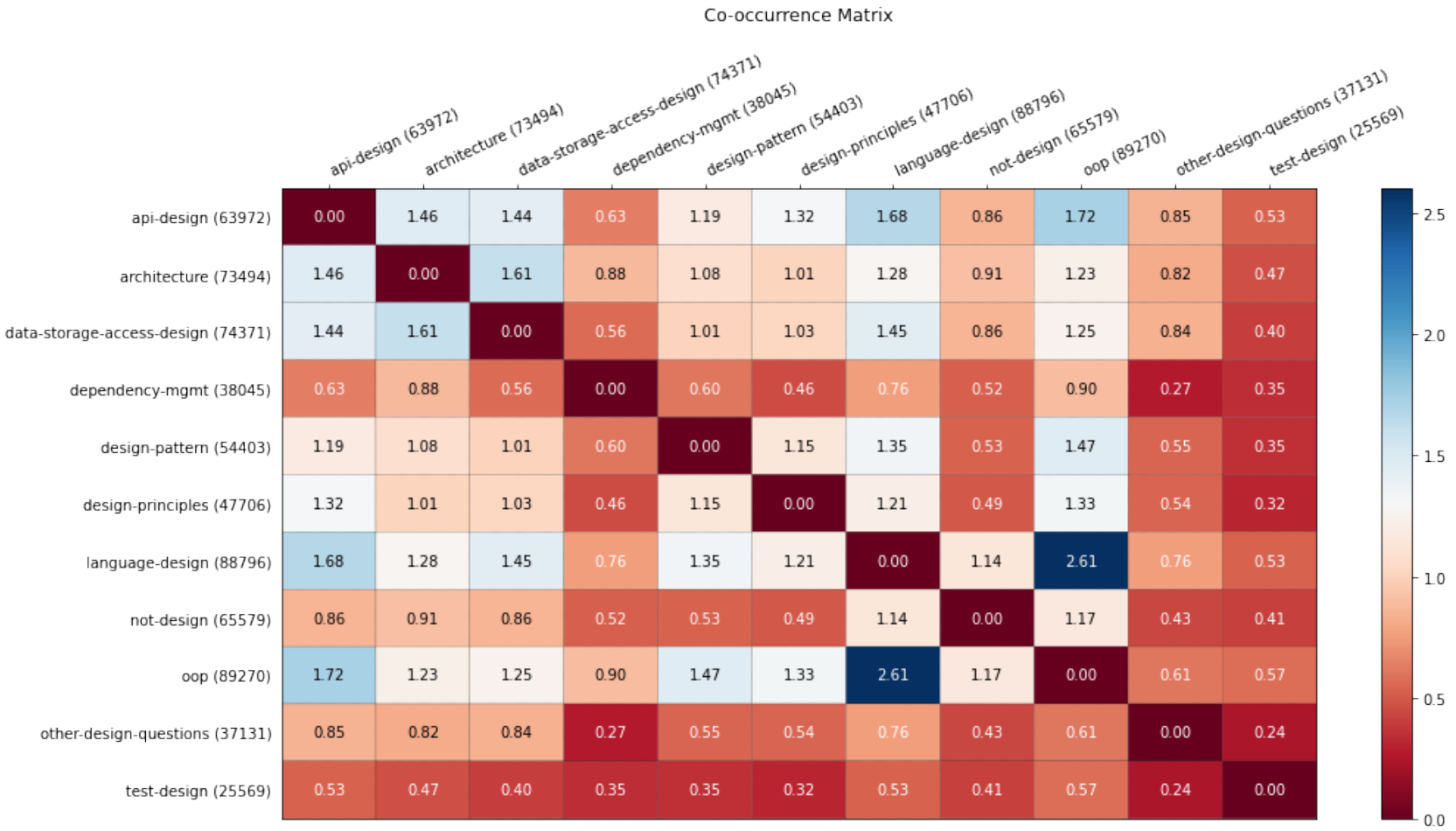

Figure 4.1: Co-occurrence matrix for the Anchored CorEx topics in Stack Overflow dataset. Values indicate the percentage co-occurrence of labels of all Stack Overflow design related posts. Higher (more blue) = more likely to co-label a Stack Overflow post.

construct level, questions being to naïve to be of significance), or to ensure that the topics identified are more non-overlapping.

4.2.3 Non-design posts tagged with design tags on Stack Overflow

As also observed during inductive coding, Anchored CorEx also identified various non-design related posts in our dataset (despite the dataset being created using only software design related tags identified using our tag filtering approach). According to Table 4.1, **65,579** of the Stack Overflow posts had some non-design related aspects in their discussions, while **10,182** Stack Overflow posts were classified purely as *not design related* (using only the not-design topic, not in combination with any other software design topic).

To investigate this further, a randomly sampled set of 50 Stack Overflow posts was sampled from the posts purely classified as not-design by Anchored CorEx. We observed that although there were very few misclassifications (2/50, posts classified as not-design even though they had some design aspects to them), a majority of the posts were related to *error resolutions* (36/50) and the remaining posts (12/50) were related to some *non-design* aspects, such as *environment setup*.

From the investigation of the randomly sampled posts, we also observed that the posts were often mistagged by Stack Overflow users (e.g. with [iterator] or [module]), even though the Stack Overflow tag descriptions of these tags state that they are meant to be used for software design-related questions. While it helps the question gain more visibility and, therefore, seems suitable for the Stack Overflow like forums, it makes identifying software design-related posts using tags more challenging. For instance, Stack Overflow's tag description for the tag [iterator] says that the tag is for questions related to the *Iterator design pattern* (a Gang of Four [27] Behavioral design pattern). But this tag was often found to be incorrectly used (e.g.,[5]) for tagging loop implementation issues.

[5] `https://stackoverflow.com/questions/39302390/`

4.2.4 Stack Overflow posts not classified under any category by Anchored CorEx

According to Table 4.1, there were **20,419** Stack Overflow posts that Anchored CorEx left unclassified (did not assign any software design topic). To investigate why these posts were left unclassified and to identify what these posts contain, we randomly sampled 100 posts from this category for manual analysis.

During the manual analysis of the posts, two thirds (67/100) of the posts were classified as *non-design*, while the remaining third of the posts (33/100) had some aspects of software design in their discussions. These misclassifications could be attributed to the sparsity of the inductively coded dataset that was used to seed the Anchored CorEx topic model.

Another observation here was that 67% of the unclassified posts had fewer than 800 characters (including spaces, punctuations and special characters; excluding code-blocks and logs) in their content, while this percentage was only 23% for the CorEx labelled posts, suggesting these posts did not have sufficient context to be effectively labelled by Anchored CorEx.

Further studies may utilize these unclassified posts in additional inductive coding rounds to effectively use Anchored CorEx for a guided exploration of the dataset.

4.2.5 Summary of findings

In this section, we presented and analyzed the distribution of the various design topics in our dataset identified using Anchored CorEx. The co-occurrence analysis identified discussion topics that were relatively discussed independently (dependency-mgmt, not-design, other-design-questions, and test-design), and the ones that had a higher co-occurrence with other design topics (api-design, language-design, and oop). For future studies, this analysis of the current view of design topics (as identified in this thesis) indicates the topics to consider and the ones to revise, for future studies aiming to identify a distribution of topics minimizing overlap.

Furthermore, the analysis of the Stack Overflow posts that were classified by the users with design-related tags (with descriptions indicating them as such) but labelled by CorEx as not-design, suggested that Stack Overflow users often use the tags for their non-design aspects (e.g. tags such as [iterator] and [module]). This indicates a need for Stack Overflow to ensure that users use the correct tagging aspects and reiterates the necessity and importance of approaches such as Anchored CorEx for

design mining over the use of Stack Overflow tags for filtering. Finally, the analysis of the posts left unclassified by Anchored CorEx, saw a third of a sampled set of posts having some design aspects in their discussions. This suggested the use of Anchored CorEx as a potential tool for a guided exploration of the dataset.

4.3 Linking GitHub references to Stack Overflow Design Topics

A study by Baltes et al. [7], identified and curated the various Stack Overflow URLs that were present in the GitHub software development artifacts (source code, commit messages etc.). In this section, we particularly study the Stack Overflow references in GitHub artifacts that refer to software design discussions. Further, we also analyze and list the various ways in which these design discussions on Stack Overflow are utilized to support open-source software development on GitHub.

4.3.1 Identification of Stack Overflow references in GitHub

To explore what design-related Stack Overflow discussions are referenced in open source projects on GitHub, we examine the *PostReferenceGH* and *GHCommits* tables available as a part of the *SOTorrent*[7] dataset. The *PostReferenceGH* and *GHCommits* tables provide references to the Stack Overflow questions, answers, and comments that are made in source code (i.e., code comments) and commit messages, respectively. We identify a subset of the Stack Overflow extended design discussion dataset posts that were referenced in GitHub repositories using these mapping tables and analyze them using both the tags used by the original poster (or modified by site moderators) as well as the labels generated by our Anchored CorEx model. We ensure that we avoid duplicate rows for both source code references and commit messages due to forking.

We found **23,634** (**6,480** after grouping by *FileId* and *PostId*) source code references and **2,251** (**221** after grouping by *CommitId* and *PostId*) commit message references on GitHub to Stack Overflow design related posts (the posts tagged with a word from our tag set). Table 4.3 shows the frequently referenced design-related Stack Overflow tags associated with Stack Overflow posts on GitHub. Stack Overflow design discussions form 0.36% and 1.13% of all source code and commit message references respectively, a high percentage of references given these posts were identified

Table 4.3: Top 10 frequently referenced tags of design-related Stack Overflow posts in GitHub. The count refers to the number of references in source code comments and commit messages that link to a design question with the associated tag on Stack Overflow.

Source Code		Commit Messages	
Tag	Count	Tag	Count
model-view-controller	9370	oop	59
oop	6353	module	23
singleton	1711	iterator	22
design-patterns	1672	language-design	17
iterator	1260	decorator	16
decorator	1080	dependency-injection	15
dependency-injection	633	singleton	14
module	622	design-patterns	13
locking	536	threadpool	10
architecture	445	locking	7

Table 4.4: Distribution of Stack Overflow references in GitHub as classified by the Anchored CorEx model. The count refers to the number of source code comments or commit messages that link to a design question with the associated CorEx topic in Stack Overflow. The Overlapping counts are the number of posts that had the label as one of the multiple assigned labels. The Exclusive counts are the number of posts that had the label as the ONLY assigned label.

CorEx Label	Source Count		Commit Count	
	Overlapping	Exclusive	Overlapping	Exclusive
api-design	5,240	74	147	0
language-design	4,508	50	128	5
oop	4,117	76	122	0
not-design	3,606	85	130	8
design-pattern	2,983	66	79	0
architecture	2,888	6	67	1
data-storage-access-design	2,612	16	71	1
design-principles	2,367	5	64	1
test-design	2,251	22	71	1
other-design-questions	2,168	20	76	2
dependency-mgmt	1,274	4	41	6
Unclassified Posts	-	150	-	10

using only 61 design related tags as compared to the 20,000+ tags on Stack Overflow that have at least 100 posts.

4.3.2 Analysis of Stack Overflow references in GitHub

Table 4.4 presents the distribution of the Stack Overflow posts referenced in GitHub as classified by the Anchored CorEx topic model. According to the user-defined tags, discussions relating to model-view-controller and oop tagged posts were most referenced in GitHub source code and commit messages, respectively. Since Anchored CorEx looks for the presence of multiple labels per reference, we observed that both the source code and commit message references shared the design-pattern label (ignoring api-design, language-design and oop; refer Section 4.2.2). Further analysis of some of the references under these categories reveals some of the interesting ways in which the GitHub projects utilize Stack Overflow posts.

4.3.2.1 Stack Overflow as a source of *Design Pattern Reference*

This category contains the GitHub code and commit messages that utilize Stack Overflow for the design pattern to be used for a particular scenario and the associated discussion around its relevance and advantages. For example, a GitHub source code[6] (shown in Figure 4.2) referenced Stack Overflow [7] discussion regarding a correct and an elegant way of implementing the *singleton design pattern* in python. CorEx classified this post as [design-pattern].

```
14  class DBConn(object):
15      """
16      Singleton object representing DB connection. Implements a big enough subset of PEP 249 for me.
17
18      singleton pattern implementation from: http://stackoverflow.com/questions/42558/python-and-the-singleton-pattern
19
20      Absolutely NOT thread safe.
21      Implemented as a singleton to minimize number of times DB connection overhead occurs.
22      Should only be created on-demand (in function) to minimize startup for other processes.
23      """
```

Figure 4.2: GitHub source code referencing a Stack Overflow post (highlighted) for the *singleton pattern* implementation.

GitHub projects also referred Stack Overflow posts for issues regarding the im-

[6] https://bit.ly/3KXmywu
[7] https://stackoverflow.com/questions/42558/

plementation of design patterns. For instance, the GitHub source code[8] (shown in Figure 4.3 references the Stack Overflow post[9] for an implementation change of the *Singleton design pattern* between PHP 5.2 and PHP 5.3. CorEx classified this post as [design-pattern].

```
19      /**
20       * Thanks to http://stackoverflow.com/questions/18907922/singleton-pattern-not-working-in-php-5-2
21       * @return type
22       */
23      public static function getInstance() {
```

Figure 4.3: GitHub source code referencing a Stack Overflow post (highlighted) for the discussion around the *singleton pattern* implementation issue in PHP 5.3.

GitHub projects were also observed to utilize the design discussions on Stack Overflow for *refactoring* the existing code implementation using design ideas presented in the Stack Overflow posts. For example, the GitHub commit message[10] (shown in Figure 4.4 references Stack Overflow post[11] for suggestions on improving the current worker threadpool implementation. CorEx classified this post as [other-design-questions].

4.3.2.2 Stack Overflow as a source for *Implementational Reference*

This category contains the GitHub code and commit messages that utilize Stack Overflow posts for low-level (implementational) references. These references often use the source code provided in the Stack Overflow post as-is or with a little modification. For instance, the GitHub source code [12] (shown in Figure 4.5 referenced the Stack Overflow post[13] for the correct design and implementation of an *Extended List* class with some modifications / additions for the target use case. CorEx classified this post as [oop,language-design].

Some of the Stack Overflow references in the GitHub projects were used as references for bug fixes and error resolutions performed. These were mostly non-design related references. For example, the GitHub source code [14] (shown in Figure 4.6

[8] https://bit.ly/3Ep1Cfz
[9] https://stackoverflow.com/questions/18907922/
[10] https://bit.ly/3vyXPbI
[11] https://stackoverflow.com/questions/19500404/
[12] https://bit.ly/36pbKZ5
[13] https://stackoverflow.com/questions/16380575/
[14] https://bit.ly/380Mhpc

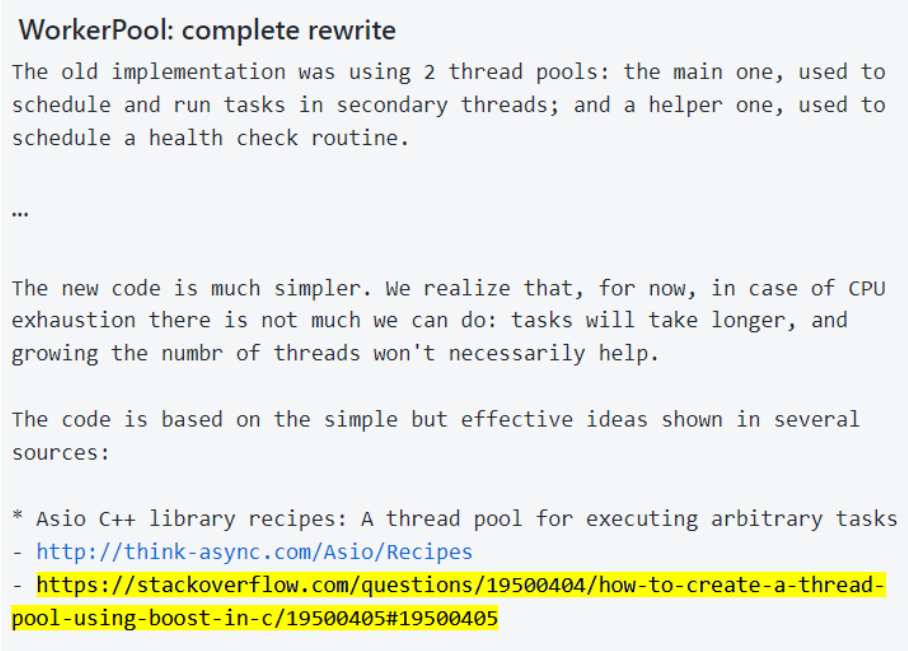

Figure 4.4: GitHub commit message referencing a Stack Overflow post (highlighted) for refactoring the existing worker threadpool implementation.

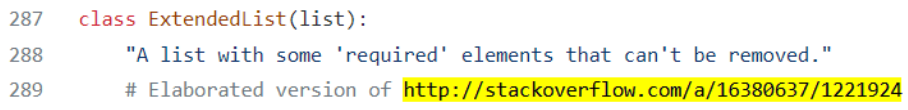

Figure 4.5: GitHub source code referencing a Stack Overflow post (highlighted) for the design and implementation of the *ExtendedList* class.

referenced Stack Overflow post[15] to fix the compilation error with a workaround due to a change in the Linux Kernel. CorEx classified this post as [not-design,test-design].

GitHub projects also referenced Stack Overflow posts for designing test cases and for addressing test related issues. For instance, the GitHub source code [16] (shown in Figure 4.7 referenced Stack Overflow post[17] to design a peculiar test case. CorEx classified this post as [test-design].

[15] https://stackoverflow.com/questions/30017344/
[16] https://bit.ly/3KW62Ns
[17] https://stackoverflow.com/questions/3511713/

```
/*
 * http://stackoverflow.com/questions/30017344/moxa-realtty-module-compilation-error-on-linux-kernel-3-16
 *
 * Function interruptible_sleep_on_timeout has been removed in kernel version 3.15.
 *
 * The workaround for the missing function can be found in many patches,
 *
 * for example [this lm-sensors patch](http://lists.lm-sensors.org/pipermail/lm-sensors/2005-February/010415.html)

 */
#if LINUX_VERSION_CODE < KERNEL_VERSION(3, 15, 0)
```

Figure 4.6: GitHub source code referencing a Stack Overflow post (highlighted) fixing the compilation error due to a change in the Linux Kernel.

```
    test("Iterant.dropLast works for infinite cursors") { implicit s =>
      // always produces at least one item
      def dropRight[A](i: Iterator[A], n: Int) = {
        // https://stackoverflow.com/a/3511985/4094860
        i.sliding(n + 1, step = 1).map(_.head)
      }
```

Figure 4.7: GitHub source code referencing a Stack Overflow post (highlighted) for a designing a peculiar test case.

4.3.2.3 Stack Overflow reference for providing *More Details / Documentation*

This category contains the GitHub code and commit messages that utilize Stack Overflow posts as a documentation reference to provide more details on the problem at hand or to corroborate the design strategy used. For example, the GitHub commit message [18] (shown in Figure 4.8 references the Stack Overflow post[19] as a documentation reference to elaborate on the problem addressed by the commit. CorEx classified this post as [api-design,oop].

Another GitHub commit message[20] (shown in Figure 4.9 referenced Stack Overflow post[21] to justify the design choice made in the commit (with the choices deliberated on in the Stack Overflow post). CorEx classified this post as [api-design,language-design,oop].

[18] https://bit.ly/3Oh2cRf
[19] https://stackoverflow.com/questions/2542747/
[20] https://bit.ly/3OfbpJL
[21] https://stackoverflow.com/questions/4630470/

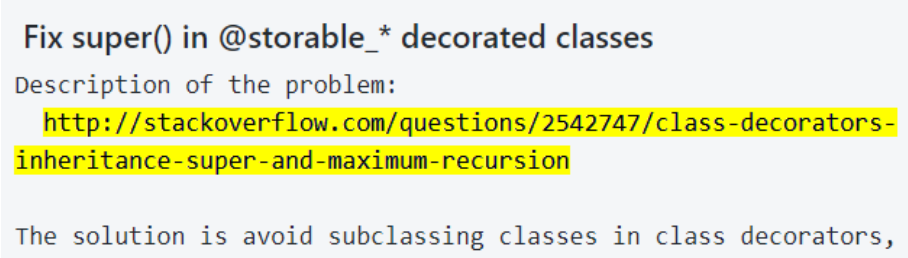

Figure 4.8: GitHub commit message referencing a Stack Overflow post (highlighted) as a source for more details on the problem being addresses.

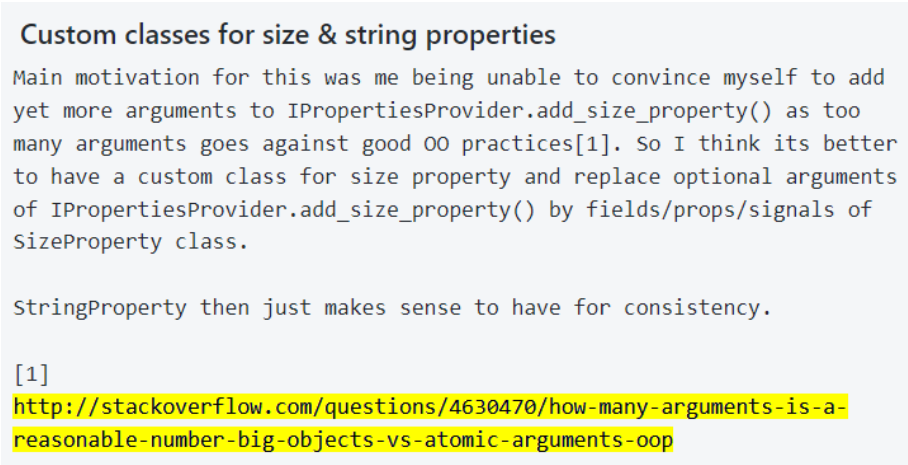

Figure 4.9: GitHub commit message referencing a Stack Overflow post (highlighted) for justifying the design choice made in the commit.

4.3.2.4 Stack Overflow references for addressing *Technical Debts / Code Smells*

This category contains the GitHub code and commit messages that list Stack Overflow posts as potential solutions to self-admitted technical debts / code smells. These posts often contain *TODO* comments highlighting the system's underlying debt. For instance, the GitHub commit message [22] (shown in Figure 4.10 references the Stack Overflow post[23] as a potential solution to use thread pools for heavy tasks. CorEx classified this post as [other-design-questions].

[22] https://bit.ly/3EosU5O
[23] https://stackoverflow.com/questions/6954584/


```
59  /**
60   * Main event loop
61   *
62   * poll(2) is portable and should be sufficient as we don't expect to handle
63   * thousands of peer connections.
64   *
65   * Initially inspired from
66   * https://www.ibm.com/support/knowledgecenter/en/ssw_i5_54/rzab6/poll.htm
67   *
68   * TODO: we could use a thread pool with pipes for dispatching heavy tasks (?)
69   * http://people.clarkson.edu/~jmatthew/cs644.archive/cs644.fa2001/proj/locksmi
70   * http://stackoverflow.com/a/6954584/421846
71   * https://github.com/Pithikos/C-Thread-Pool/blob/master/thpool.c
72   */
73  bool server_run(const struct config *conf)
```

Figure 4.10: GitHub source code referencing a Stack Overflow post (highlighted) as a potential solution for the self-admitted technical debt. The red square highlights the self-admitted technical debt.

4.3.3 Summary of findings

In this section, in an attempt to demonstrate the usefulness of the design knowledge on Stack Overflow for software development, we analyzed the various references that GitHub projects made to Stack Overflow design posts. We identified that 0.36% and 1.13% of Stack Overflow references in GitHub source code and commit messages respectively, were to design tagged posts. Furthermore, we identified four ways in which these design references are used. We found that these projects used Stack Overflow as a source of design patterns and implementational references, as they contained context- or technology-specific discussions. Additionally, the Stack Overflow references were also used as documentation references for the design choices made or as potential solutions to the self-identified technical debts and code smells. Design mining studies such as ours might help make design knowledge on Stack Overflow more democratized, by making it easily searchable and accessible. It could additionally help in the development of tools that leverage Stack Overflow to improve design documentation and validation.

4.4 Chapter Summary

In this chapter, we utilized the design knowledge identified using the topic modelling technique (Anchored CorEx) arrived at in the previous chapter for addressing **RQ-2**

(How can we leverage the knowledge of design topics discussed on Stack Overflow?). First, we leveraged Anchored CorEx to gain insight into design discussions on Stack Overflow and to validate the effectiveness of our classification approach. We observed that dependency-mgmt, not-design, other-design-questions and test-design had the least co-occurrence with other design topics, being more focused and having very few latent terms associated with other topics. We also identified api-design, language-design and oop to not be good classifiers of software design, given their significant overlap with other design topics.

Further, despite our tag-filtering for design tags, we observed a significant presence of non-design-related posts in our dataset. On further analysis, we found that Stack Overflow tags, such as [iterator] or [module], had non-design-related posts, speculated to be added by users to increase the visibility of the post. Additionally, we also analyzed posts left unclassified by Anchored CorEx and attributed the reason for them being as such to the sparsity of the inductively coded dataset or the lack of context in those posts.

Finally, our analysis of the Stack Overflow posts referenced on GitHub led us to identify the various ways in which design discussions on Stack Overflow played an important role in open source projects on GitHub. We found that Stack Overflow posts were used for design pattern and implementational references in GitHub projects. Additionally, the Stack Overflow posts were also used for providing documentation references for elaborating on the design challenges or for justifying the design choices. The Stack Overflow posts were also referenced for documenting potential solutions to technical debts and code-smells. Identifying these use-cases of the Stack Overflow design discussions in GitHub projects could help in the development of tools that may help ease this process. We will discuss this and other potential solutions in more detail in the next chapter.

Chapter 5

Discussion, Limitations & Implications

5.1 Introduction

We began this thesis with an analysis of the various approaches for design topic classification of Stack Overflow posts. In this chapter, we first begin by extending this discussion by comparing the various approaches. In Section 5.2.1, we discuss the need to perform inductive coding despite the presence of the user-defined Stack Overflow tags. Next, in Section 5.2.2 we discuss how the two automated topic modelling approached discussed in this thesis (LDA and Anchored CorEx) stack up against each other. Further, in Section 5.2.3 we provide a detailed discussion of the evaluation metrics used and argue for the need for a balanced approach to design mining.

Furthermore, having established the advantages of using a semi-supervised approach for topic modelling, and to foster its use for further software engineering research, in Section 5.3 we provide some guidelines for performing semi-supervised topic modelling in SE. Additionally, based on our experience, for future studies specifically seeking design knowledge, in Section 5.4 we discuss some of the advantages and disadvantages of sourcing this knowledge from Stack Overflow. We then report on the threats to the validity of the research presented in this thesis and the steps we have taken to mitigate these threats (Section 5.5).

In the second part of this thesis, we discussed how the design knowledge on Stack Overflow could be leveraged to gain insights into the design discussions on Stack Overflow, and its utility for software development by analyzing Stack Overflow references

Table 5.1: Example Stack Overflow questions, and approaches to identify latent topics

Question Id & Title	Inductive Code	Stack Overflow Tags	CorEx Topics
11242144: *Optimal TPL Dataflow Design*	architecture	[c#, architecture, asynchronous, concurrency, tpl-dataflow]	[api-design, architecture, design-principles, language-design]
34258867: *How factory pattern works*	design-pattern	[php, oop, factory]	[design-pattern, language-design, oop]

in open source GitHub projects. Therefore, we conclude this chapter by discussing some of the ways in which this study could be extended further, some implications for researchers and practitioners, and some potential solutions that our study enables (Section 5.6).

5.2 Un-/Semi-/Fully-Supervised Approaches

This thesis involved the use of various classification/topic modelling approaches for classifying software design discussions. We used a manual approach (inductive coding), an unsupervised approach (LDA), and a semi-supervised approach (Anchored CorEx). In this section, we discuss our research design choices with respect to the various approaches used, some arguments to justify those choices, and our observations. To do so, we additionally use a fourth topic set created using the user-assigned *tags* (tagset introduced in Section 3.2.1) on the Stack Overflow question. All four approaches label the content of the Stack Overflow post with respect to some aspect of software design. Table 5.1 shows the Stack Overflow question title, the inductive code we assigned, the user-defined tags (where *user* is the asker of the question on Stack Overflow and the possible editors of those tags), and the set of associated Anchored CorEx topics. Here we see that the CorEx Topics closely reflect the inductive codes as well as the Stack Overflow user defined tags. We also observe CorEx additionally also assigns certain topics [language-design] that we identified to be spurious correlations (as discussed in Section 4.2.2), that should be ignored. Table 5.2 captures the summary of the benefits and drawbacks of each of the approaches. The results discussed in this thesis corroborate these findings.

Table 5.2: Four approaches to categorizing latent design topics.

Approach	Pros	Cons
LDA	Efficient, tunable. Assign many (all) topics to a question [13].	Potentially unstable; topics hard to explain; sensitive to dictionary [3, 57, 56]. Does not model correlations [32].
Anchored CorEx	Improved recall on LDA, efficient, scalable. Allows for expert input (semi-supervision) making topics more interpretable [62, 26]. Models for correlations.	Lower precision. Possibly unstable: subject to bias from supervision.
Inductive Coding	Qualitative and expertise-based [50], handles nuance.	Not scalable; sensitive to sample; coder agreement low.
Stack Overflow Tags	Crowd-sourced, specific.	Edited by others; not all design.

5.2.1 On the need for Inductive Coding over using user defined Stack Overflow Tags

We leveraged the existing tags on Stack Overflow to generate our design Q&A dataset. We then inductively coded this dataset. One might ask why the coding process was even necessary, given the users had already defined tags. Our inductive coding process is high-effort, but we showed that many posts tagged as [design] were in fact not design related (see Section 4.2). Additionally, since the Stack Overflow tags are created by users, they only provide a general sense of the question being asked and do not have a strict binding to be related to design aspects (despite the tag descriptions).

Furthermore, tagging in Stack Overflow is contentious, including for the reasons Barua et al. [9] mention: minor syntactic differences, tag evolution, and tag semantics (e.g., the design pattern vs. design principle may be perceived differently by different people). More importantly, manual coding allows us to thematically group posts and identify broader patterns (such as the motivation for asking the question). Additionally, we were able to use this inductively coded dataset to seed the Anchored CorEx approach, which is much more scalable.

5.2.2 On the differences between LDA and Anchored CorEx

The output of LDA reflects a statistical generative model that may be hard to interpret for a more abstract concept like software design. This results in issues such as topics identified not being according to the area of interest, causing the resultant distribution to be less useful (e.g., combining databases and architecture, while concurrency gets its own topic). Additionally, this could also cause manual labelling of LDA topics to be biased towards the area of interest (in our case design topics), whereas in reality what it represents may be very hard to interpret (even when using the top indicative terms or analysis of sampled posts). This is a tricky and subjective aspect of LDA [33]. With CorEx, the topic labels come from the anchor set a priori and there is no labelling topics post-hoc. This results in CorEx topics being much more interpretable because they can be tied back to the domain knowledge identified during inductive coding. The preliminary results by studies such as the one by Reing et al. [62] also confirm that the Anchored CorEx approach results in topics that are precise and interpretable. Future studies involving multiple domain experts performing an in-depth analysis of the CorEx topics may confirm these findings.

Furthermore, LDA does not model topic correlations. Since LDA uses a conjugate Dirichlet prior, it almost independently models topic occurrences and fails to capture topical correlations [32]. Anchored CorEx, by contrast, models for correlations in the data by searching for latent factors that minimize them [77]. It, therefore, enables performing rich analysis providing additional insights such as the co-occurrence analysis in Section 4.2.2 of this thesis.

Finally, design labelling (and approaches such as inductive coding) in general is imperfect at the human level, which suggests tools will also struggle without better definitions. We see CorEx as providing a useful way, with anchoring, to more precisely define design concepts, training the topic model with less common or generic topics, which LDA suffered from.

5.2.3 On the need of a balanced approach for design mining

While LDA and Anchored CorEx provide similar performance in terms of their coherence scores (see Section 3.4), we observe that while CorEx has a much higher recall (R) and balanced accuracy (AUC), it had a similar precision (P) as LDA in at least 2 of the 4 cases. Are these the correct metrics to be used as the basis of our evaluation? Is it a problem that CorEx has a higher recall, but lower precision, in general? In

this section, we discuss some of the questions surrounding these metrics and argue for the need for a balanced approach for design mining.

P / R is a trade-off. In some applications, it may be better to get all relevant questions instead of focusing on precision, thus increasing precision at the cost of recall may not be appropriate [11]. Our evaluation is exploratory; for example, it is possible users are missing a label which should really be there (i.e., users may be mistaken). Therefore our case having a higher recall would be more beneficial than aiming for higher precision.

Additionally, even though Anchored CorEx had a higher recall than precision, having a high balanced accuracy indicated that it rejected the negative cases well, especially in our case with imbalanced data. Since balanced accuracy is the average of the True Positive Rate (TPR; Sensitivity; the percentage of positive cases the model is able to identify) and the True Negative Rate (TNR; Specificity; the percentage of negative cases the model is able to identify), it considers the negative examples the model rejected well, and is, therefore, a better metric than F1 score (the harmonic mean of precision and recall) for imbalanced datasets.

Further, our definition of a true positive may be overly constrained. Just because our two coders tagged a question as 'architecture' does not imply the original question was about architecture; conversely, the user tag might indeed be architecture, but at a lower abstraction level (such as 'hexagonal').

One of the advantages of LDA-style approaches is that they are objective (to some degree; choice of hyperparameters and dataset order affect the reliability [3]). Inductive coding, although rigorous [50], depends on the skill and experience of the coders. Yet we have shown that *some* supervision greatly increases topic relevance. This suggests that a balance is important. In this thesis, some aspects were objective: (randomly) selecting sample data; running LDA; selecting hyper-parameters; and running CorEx. Some were subjective: inductively coded data; naming LDA topics; selecting anchor words using c-TF-IDF. We argue that **this balance is inherent in SE domains, such as design, which are not objectively defined themselves**.

For example, while selecting anchor terms, we cannot simply remove terms, because those terms might well be relevant, and it is our knowledge of the topic that is lacking. On the other hand, as Agrawal et al. have reported [3], repeatability and stability in topic modelling are also important. Ideally, the topics Anchored CorEx identified in this study should be relevant in another study utilizing the same data.

5.3 Guidelines for semi-supervised topic modelling in SE.

Having established Anchored CorEx as the topic modelling approach and the need for a balanced approach for mining software design, in this section we present some guidelines for using semi-supervised topic modelling in SE, based on our experiences.

1. **Clearly specify criteria to avoid topic overlap.** To ensure clear topic separation, merge highly overlapping topics. Identifying overlapping topics while preparing the initial training set may be difficult. Therefore, approaches such as co-occurrence analysis (refer Section 4.2.2) can be employed to identify overlapping topics. The use of hierarchical topic modelling could also help topic separation.

2. **Filter out posts/documents** using (a) experience criteria for the author of the post; (b) exclusion criteria for code snippets and error logs; (c) exclusion criteria for questions that do not have a selected answer. These criteria would ensure that the dataset being modelled contains only highly relevant posts, thus reducing spurious topic-word correlations and requiring fewer iterations of Anchored CorEx analysis to remove those spurious terms. In this study, we used our tag filtering approach along with the criteria of the Stack Overflow tags having a minimum of 100 posts as our filtering criteria. Being an exploratory study, we chose not to use very strict filtering criteria to avoid missing out on design topics.

3. **Clearly define a goal for the resultant topic model.** This would help to streamline the topic modelling activity and the definition of the anchor terms. For example, Soliman et al.'s investigation into *middleware* [70]. The goal of our topic modelling was to identify design topics on Stack Overflow. This helped pick relevant terms from our c-TF-IDF analysis to be used as anchor terms and weed out irrelevant and spurious ones.

4. **Experiment with various sampling frames and document granularities** (document, paragraph, sentence). A more granular post type may improve precision, but may also require more effort to scale to a large sampling frame. For example, Viviani et al. [79] chose paragraph as their level of granularity to localize design points. In this study, we chose a Stack Overflow question along

with its associated answers as our level of granularity. We discuss more on this choice in Section 1.3.

5. **For design mining, consider the scale of the design solution** (method level, file level, program/architecture level). For example, Ernst et al. [23] presented various levels of quality analysis and chose the upper 2 levels as corresponding to design. Having a well-defined taxonomical guide or a decision tree would help reduce subjectivity and also avoid confusion in case manual coding is employed. In this thesis, we used a coding guide to provide the coders with a sense of scale when performing inductive coding. Further, we identified the confusion due to scale as a *Type 6* disagreement (see Section 3.3.1.1).

6. **Suggested researcher degrees of freedom when performing anchoring for semi-supervised topic modelling.** When performing anchoring for semi-supervised topic modelling, (a) experiment with various seed words, anchor strength and topic numbers; (b) after every iteration manually access overlapping terms and anchor them as needed; (c) use strategies such as c-TF-IDF to glean potential anchor words; (d) refer to anchoring strategies by Gallagher et al. [26] for topic separability, representation, and aspects.

5.4 Stack Overflow as a potential source of design knowledge

In this thesis, we chose Stack Overflow as a basis for analyzing design discussions, for it being unparalleled to any other platform when it comes to the breadth of the software discussion topics that it hosts or its popularity among software developers [7]. In Section 2.2, we discussed the relevance of Stack Overflow for software design. We then confirmed the presence of design discussions on Stack Overflow and identified its various types. In this section, we discuss the unique advantages and challenges that are associated with using Stack Overflow as a source of design knowledge, that we specifically observed and identified while performing this research.

5.4.1 Advantages of using design knowledge on Stack Overflow

In addition to some of the ways in which Stack Overflow references help open source developers that we discussed in Section 4.3, in this section we present some of the additional advantages of Stack Overflow design references that we observed.

First, we observed that some design-related questions were highly discussed and also sparked or linked to discussions on other platforms such as blogs[1]. This highlights the multi-platform nature of design discussions on Stack Overflow. This multi-platform nature of Stack Overflow design discussions not only provides credibility and context to the discussion, but it also (given the popularity of the platform) represents a high coverage of all the design discussion areas.

Further, while some questions discussed the existing design choices[2], some even extended this discussion on how to work with the existing design choices of the system to achieve a particular goal[3]. Since most of the questions posted on Stack Overflow were not limited to any particular project, the developers answering these questions were not limited by the existing design choices or the requirements of any such projects (an advantage not available in project specific platforms). This represents that the discussions posted on Stack Overflow could be both context specific (applicable to a specific scenario, system architecture/design) or context independent (widely applicable).

The developers answering the questions were free to discuss multiple design approaches while also documenting the advantages and the disadvantages of each[4]. Moreover, Stack Overflow posts discussing high-level design patterns (e.g. [5]) were not limited to specific programming languages and were tagged with multiple programming language tags that have a similar underlying language design. This allowed the developers to get the support of a wider community.

[1] https://stackoverflow.com/questions/28521381/
[2] https://stackoverflow.com/questions/547710/
[3] https://stackoverflow.com/questions/1986152/
[4] https://stackoverflow.com/questions/17466150/
[5] https://stackoverflow.com/questions/42687416/

5.4.2 Potential challenges for developers using the design knowledge on Stack Overflow

While Stack Overflow provides a bouquet of benefits for developers looking for design knowledge, as discussed in the previous section, it also comes with its own set of unique challenges.

Stack Overflow does not currently have a single all-encompassing tag to identify all the design-related questions (the Stack Overflow tag [software-design] is not frequently used and is used only for a small fraction of the design related questions). This is because Stack Overflow was not designed specifically to be used for discussing software design, but as a general Q&A platform for developers. Additionally, this is also because software design encompasses a broad range of topics from software architecture and design patterns, to proper use of language idioms. While the specificity of topics may be beneficial to the Stack Overflow use case, the lack of an overall view of software design presents a hurdle when trying to utilize the wealth of knowledge contained in Stack Overflow for building software design-related tools.

This leads to the use of the existing Stack Overflow user-defined tags being the best alternative for identifying software design related questions. But as the results of this thesis (and several others, e.g. [45]) highlight, these tags are either incorrectly used, or are used referring to some non-design aspect of the tag label, resulting in the set of posts identified by the tags not all being related to design.

Further, the context specificity of the posts or the lack of it thereof, may result in multiple Stack Overflow posts discussing the same design challenge. Stack Overflow tries to alleviate this problem by marking duplicate questions and suggesting similar questions (right from the time when a user is about to ask a question, to a user exploring posts). Despite this, users often need to go through multiple posts to get all possible alternative solutions, or to get solutions pertaining to their specific context. The context related challenges can be addressed by using software design context mining (such as the one presented by Viviani et al. [78]) or automated search tools (such as the one presented by Ponzanelli et al. [60]).

Finally, users exploring the Stack Overflow posts require additional efforts in order to validate the design solutions presented. This is often done by either reviewing or taking part in deliberations in the question or answer comments, or by establishing the reputation of the user asking the question, or through upvotes.

A majority of these challenges can be addressed by using automated design mining

tools/methods/datasets, like the ones presented in this thesis. These challenges also present some opportunities for research and building tools for software design and development (we discuss some such opportunities in Section 5.6).

5.5 Threats to Validity

There are several threats to the validity for the work presented in this thesis. In this section, we summarize the threats and also present the steps taken to mitigate them. For repeatability and reliability purposes, we provide the datasets used, the analysis notebooks, and the results of the various phases, in our replication package [1].

5.5.1 Internal Validity

We filter our dataset using a set of design tags; therefore, if a design discussion uses a tag not in that set, we would miss this as a potential source of data. To mitigate this we conducted several rounds of tag set expansion (see Section 3.2.1). Barua et al. [9] highlight some issues, including problems with minor, syntactic changes in tags that do not affect semantics, and tags which are quite generic. In our case, the tag [design-pattern] seems to occur even when the question is not specific to a known pattern like *Singleton*. Tags also evolve, and indeed, Stack Overflow moderators recently completed a major refactoring of the previous [design] tag, re-tagging posts with more specific tags.

Further, since the dataset relies on a limited set of human labelled data (or makes the possibly invalid assumption that the tagging in Stack Overflow reflects real design). However, we reached code saturation, and we observe that the posts classified by Anchored CorEx, the LDA topics and the human codes are quite similar. Thus, we do not believe we missed any substantive codes.

Our anchored/supervised topic model approach using CorEx relied, for some topics, on fewer than 10 *documents* (Stack Overflow question/answer sets). This means the topic model had limited supervision on these topics and this might limit the applicability of the matched questions. However, we reported recall, precision, F1 score, and balanced accuracy to show how well the anchoring worked, even with limited training.

Since we had poor agreement between coders, one might argue our coding dictionary was poorly constructed. We believe this was not the main problem, and

rather, the problem is that design in software is poorly defined. Consider the distinction between architecture and design. Conventionally one might place architecture as 'higher' in the abstraction hierarchy for a software system, pertaining to a wider scope of impact, and more modules affected. But this distinction remains unclear, and many use the terms interchangeably. Devising a coding scheme that can delineate design concepts is tricky. And yet many of the Stack Overflow questions we examined were clearly in one category or the other, suggesting there is a difference.

Our true positives for precision, recall, F1 score and balanced accuracy were based on alignment between user tags and our topic names. This was a reasonable choice to explore the differences but a more sophisticated metric will be more informative.

The analysis of the uses of the Stack Overflow references in GitHub projects was based on the results of tag-filtering and Anchored CorEx. This approach would have likely missed out other uses of the Stack Overflow references, for the design tags that were not a part of our tag set or for the tag being incorrectly labelled by Anchored CorEx. This risk would have likely been alleviated by the rigor of the two approaches as reported earlier.

5.5.2 External Validity/Sampling Frame

The objective of our thesis is to identify various software design related questions asked and the software design related challenges faced by developers, as discussed on public question and answer forums. The theoretical population that this thesis is concerned with are the design topics discussed (i.e., in a question or answer) by software developers on public Q&A forums. For this we purposively choose the design related questions asked (using our tagsets) on Stack Overflow as our sampling frame. Our results should generalize to our chosen sampling frame. Other studies utilizing a different sampling frame may find design topics relevant to the type of the forum being studied. We examined GitHub Discussions as another potential source, but found there were very few discussions about design.

Further, we analyzed the various use of the Stack Overflow references. For this we purposively analyzed the GitHub references to design tagged Stack Overflow questions from our tag set. Analysis of Stack Overflow design references in other software development, communication and collaboration platforms may reveal other potential use cases of the references.

As a qualitative study, since we present our results with respect to a specific

context, we believe the lack of generalizability to not be a critical threat to the validity of the results presented in this study.

5.5.3 Construct Validity

The taxonomy of software design discussions on Stack Overflow presented in this thesis relies on our view of the software design discussions. Other approaches for classifying design discussions (such as the motivation of the asker for initiating the discussion) might result in a different taxonomy. The classification of design presented in this thesis relies on our understanding of design, as well as the results of the topic modelling approaches. Further, we present our results using standard topic modelling approaches and a newer Anchored CorEx approach. It is possible that using some other approaches could improve results.

5.5.4 Ethical Considerations

Stack Overflow is a public website and content created there is acceptable for research according to the Stack Overflow licence[6]. GitHub defines Acceptable Use Policies we adhere to.[7] However, while our study looked at publicly available archival data protected by law, and reports on non-personal information, ethical considerations including privacy concerns and possible harm, such as reputational damage, are relevant, should we expand the scope of the research to, for example, consider individual questions. At that point, informed consent should be obtained [29].

5.6 Practical Implications & Future Work

The results of our study have some implications for both researchers and practitioners. For **researchers**, we build a corpus of design questions for future tools to refer to, for example, for fine tuning word vectors. Further, our results could help improve software documentation knowledge and provide a better perspective on how software design activity happens.

For **practitioners**, we believe Stack Overflow holds considerable value as a dynamic corpus of design knowledge. The methodologies discussed in this study could

[6] `https://stackoverflow.com/legal/terms-of-service#access`

[7] `https://docs.github.com/en/github/site-policy/github-acceptable-use-policies#5-information-usage-restrictions`

help make this knowledge more accessible to be used for building, for example, a recommender system for similar questions or design problems, for improving documentation systems, or for improving Stack Overflow posts labelling (like in Bangash et al. [8]) to better direct those posts to people with the relevant knowledge. In this section, we discuss some such solutions that we believe would most benefit from the findings of this study.

5.6.1 Improving design classification

We first discuss the various ways in which future studies could extend our work, by analyzing the design classification problem discussed in this thesis using different approaches and perspectives. While we present one perspective of classifying design discussions, future studies could choose to analyze the corpus of Stack Overflow design discussions using other perspectives such as *reason for asking the question*, *non-functional design requirements discussed*, or the *context-specificity of the questions*. These various perspectives could then be analyzed together using a qualitative *Grounded Theory* [28, 34] like approach, to not only achieve a better classification, but also to gain better insights to the various socio-technical aspects of software design discussions.

Further, future research could also look at other platforms frequented by software developers for their software design related queries (such as Software Engineering Stack Exchange, Reddit, Blogs, etc.). Analyzing these platforms along with the multi-platform nature of software design discussions can help in delineating the software design reference platforms used, for improving search, and for gaining insights into the crowd sourcing aspects of software development.

5.6.2 Identifying changing design approaches

Software design is an ever-changing field that keeps evolving with the evolution of technology, languages, and frameworks. This leads to the creation of new design patterns or the depreciation of some old ones. Stack Overflow, being a popular software development related Q&A platform, closely mirror this evolution. One potential application of this evolving design knowledge on Stack Overflow is the preservation of this knowledge through the development of new or updating the existing static analysis rules for code inspection/reviews. For instance, with the introduction of React Framework, a plethora of new design patterns were introduced such as Redux

and Flux, which were considered to be an evolution over the preexisting MVC design pattern. Stack Overflow mirrored this evolution and consisted of many questions addressing the common issues and challenges with these frameworks. This knowledge could be preserved/applied in the form of static analysis rules to improve code reviews. Further, with the evolution of the React Framework, a new feature called 'hooks' was introduced, which obsoleted the use of the Container-View Pattern. Further, we speculate design discussions platforms such as Stack Overflow, being highly democratized, to be the prime sources of design problems for the software development community's consideration.

5.6.3 Validating ML-based code generation

With the advancements in generative machine learning models (such as GPT-3) and the availability of massive amounts of compute power, have led to major improvements in automatic code generation tools such as GitHub Copilot[8]. Since these tools take advantages of large-scale public code repositories for training, the code that they generate often represent the best guesses for what most people would write, rather than suggesting code with the correct design. The incorrect code (if) generated appears deceivingly valid and is often accepted by programmers due to their automation and anchoring biases[9]. Stack Overflow design knowledge could be leveraged here by either validating the code generated or by enforcing the pattern for code generation.

5.6.4 Recommending related design discussions

Alternative to automated code generation, design knowledge could be leveraged to review the code changes before there are merged into the shared repository code to highlight design flaws/anti-patterns and to suggest relevant Stack Overflow posts wherein such flaws have been detected. One of the alarming observations made by Zanaty et al. [83] states that design concerns are not commonly discussed in code reviews. And in cases where such discussions take place, only a small percentage of them elicit the author's response. Therefore automated identification of design flaws can foster such a discussion and the suggestion of relevant Stack Overflow posts can reduce design flaws and the build up of technical debt.

[8]https://copilot.github.com/
[9]https://www.fast.ai/2021/07/19/copilot/

5.6.5 Automating design knowledge documentation

In Section 4.3, we observed that roughly 1% of GitHub-linked discussions were design-related. We could use this dataset for both generating documentation (e.g., on-demand as someone reads the code) and adding documentation sources for writers. One use case, for instance, could be to retrieve Stack Overflow posts given source code similarity. Something like this is shown by Xu et al. [80] for code generation.

Soliman et al. [70] found a similar phenomenon with architecture knowledge for middleware. It may be possible to add automation to this and suggest relevant Stack Overflow style discussions as source code comments. Such simple links would be considerably simpler to document than the entire design choices, much like naming a class SingletonX implicitly links to the Singleton design pattern (for those who have learned that pattern).

5.7 Chapter Summary

In this chapter, we reflected on the analysis and results of the various phases of the research study and discussed our findings and implications for future studies. First, we began with a comparison of the various design topic classification approaches discussed in this thesis. We argued for the need for performing inductive coding over the use of the available user-defined Stack Overflow tags, due to the tags not correctly reflecting the design aspects of the questions, and for the advantages that the inductive coding approach provides such as the flexibility for thematically grouping the posts and identifying broader patterns. We then compared the unsupervised (LDA) and semi-supervised topic modelling (Anchored CorEx) approaches and observed Anchored CorEx to be a better approach for being able to better define design topics and for modelling correlations. We then compared the various metrics used in our evaluation and suggested balanced accuracy and recall to be the better indicators of performance in or case of design classification with imbalanced datasets. Additionally, we also argued for a balanced approach for design mining, one that balances subjectivity for model performance, with semi-supervised modelling using the Anchored CorEx approach being the prime candidate.

Furthermore, we provided some guidelines for future studies performing semi-supervised topic modelling. Additionally, based on our experiences in this study, we recommended using Stack Overflow as a potential repository of design knowledge and

provided the advantages (such as multi-platform nature, having a high coverage of design aspects, having both context-specific and context-independent questions, etc.) and challenges (such as being difficult/tedious to identify and validate etc.) of using this knowledge.

Finally, we discussed the threats to the validity of our research and the steps taken to address those threats. We concluded by presenting the implications of our study for researchers and practitioners, and some potential solutions enabled by our findings.

Chapter 6

Conclusion

Software design is an inherently subjective, yet important aspect of software development. After building a design-related Stack Overflow dataset, we used an inductive coding method to identify the design-related topics/areas discussed on Stack Overflow. We found it difficult to get inter-coder agreement on certain design topics and list some possible reasons. We then used two scalable techniques, LDA and Anchored CorEx, to characterize the topics in the dataset. We found that the semi-supervised Anchored CorEx approach was more interpretable and produced more coherent topics. It outperformed LDA on recall and balanced accuracy metrics, the metrics we discussed to be most suitable for imbalanced datasets.

Further, we establish the usefulness of the resulting topic model by exploring two of its practical applications. First, we explore the design-related topics discussed on Stack Overflow. This also helps us establish the relevance of the inductively coded labels, identify challenges with using the user-defined Stack Overflow tags, and review the effectiveness of the Anchored CorEx approach. Second, we qualitatively identify four ways in which open-source GitHub projects leverage Stack Overflow design discussions.

Based on our experiences in this study and our findings, we present guidelines for future studies, to foster the use of semi-supervised topic modelling in SE. We additionally also discuss the advantages and challenges for using Stack Overflow as a design knowledge repository. Our experience using the semi-supervised CorEx approach leads us to believe that approaches like CorEx that combine domain knowledge and scalability are key for analyzing SE text repositories, particularly those with subjective latent topics like design.

Bibliography

[1] Characterizing Design Discussions With Semi-Supervised Topic Modeling. `https://doi.org/10.5281/zenodo.5885783`, January 2022.

[2] Rabe Abdalkareem, Emad Shihab, and Juergen Rilling. What do developers use the crowd for? a study using stack overflow. *IEEE Software*, 34(2):53–60, 2017.

[3] Amritanshu Agrawal, Wei Fu, and Tim Menzies. What is wrong with topic modeling? and how to fix it using search-based software engineering. *Information and Software Technology*, 98:74–88, June 2018.

[4] Mubashir Ali, Husnain Mushtaq, Muhammad B Rasheed, Anees Baqir, and Thamer Alquthami. Mining software architecture knowledge: Classifying stack overflow posts using machine learning. *Concurrency and Computation: Practice and Experience*, March 2021.

[5] Rana Alkadhi, Teodora Lata, Emitza Guzmany, and Bernd Bruegge. Rationale in development chat messages: an exploratory study. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 436–446. IEEE, 2017.

[6] Vimala Balakrishnan and Ethel Lloyd-Yemoh. Stemming and lemmatization: A comparison of retrieval performances. In *Proceedings of SCEI Seoul Conferences*, pages 625–633, 2014.

[7] Sebastian Baltes, Lorik Dumani, Christoph Treude, and Stephan Diehl. Sotorrent: reconstructing and analyzing the evolution of stack overflow posts. In Andy Zaidman, Yasutaka Kamei, and Emily Hill, editors, *Proceedings of the 15th International Conference on Mining Software Repositories, MSR 2018, Gothenburg, Sweden, May 28-29, 2018*, pages 319–330. ACM, 2018.

[8] Abdul Ali Bangash, Hareem Sahar, Shaiful Chowdhury, Alexander William Wong, Abram Hindle, and Karim Ali. What do Developers Know About Machine Learning: A Study of ML Discussions on StackOverflow. In *International Conference on Mining Software Repositories (MSR)*, pages 260–264, Montreal, QC, Canada, May 2019.

[9] Anton Barua, Stephen W. Thomas, and Ahmed E. Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering*, 19(3):619–654, November 2012.

[10] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. SEI Series in Software Engineering. Addison-Wesley Professional, 3rd edition, 2012.

[11] Daniel M. Berry, Ricardo Gacitúa, Peter Sawyer, and Sri Fatimah Tjong. The case for dumb requirements engineering tools. In *International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*, 2012.

[12] Tingting Bi, Peng Liang, and Antony Tang. Architecture patterns, quality attributes, and design contexts: How developers design with them. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, pages 49–58, 2018.

[13] David M Blei and John D Lafferty. Topic models. In *Text mining*, pages 101–124. Chapman and Hall/CRC, 2009.

[14] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[15] João Brunet, Gail C Murphy, Ricardo Terra, Jorge Figueiredo, and Dalton Serey. Do developers discuss design? In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 340–343, 2014.

[16] Joshua Charles Campbell, Abram Hindle, and Eleni Stroulia. Latent dirichlet allocation. In *The Art and Science of Analyzing Software Data*, pages 139–159. Elsevier, 2015.

[17] Rafael Capilla, Anton Jansen, Antony Tang, Paris Avgeriou, and Muhammad Ali Babar. 10 years of software architecture knowledge management: Practice and future. *Journal of Systems and Software*, 116:191–205, June 2016.

[18] Tse-Hsun Chen, Stephen W Thomas, and Ahmed E Hassan. A survey on the use of topic models when mining software repositories. *Empirical Software Engineering*, 21(5):1843–1919, 2016.

[19] Xingyu Ken Chen, Anais Ang, Jing Yi Lee, Jason Wong, Neo Loo Seng, Gabriel Ong, and Majeed Khader. Identifying the public's psychological concerns in response to covid-19 risk messages in singapore. *Journal of International Crisis and Risk Communication Research*, 4(2):309–334, 2021.

[20] Paul Clements, David Garlan, Reed Little, Robert Nord, and Judith Stafford. Documenting software architectures: views and beyond. In *25th International Conference on Software Engineering, 2003. Proceedings.*, pages 740–741. IEEE, 2003.

[21] Filipe R Cogo, Xin Xia, and Ahmed E Hassan. Assessing the alignment between the information needs of developers and the documentation of programming languages: A case study on rust. *arXiv preprint arXiv:2202.04431*, 2022.

[22] Osama Ehsan, Safwat Hassan, Mariam El Mezouar, and Ying Zou. An empirical study of developer discussions in the gitter platform. *ACM Transactions on Software Engineering and Methodology*, 30(1):1–39, January 2021.

[23] Neil A Ernst, Stephany Bellomo, Ipek Ozkaya, and Robert L Nord. What to fix? distinguishing between design and non-design rules in automated tools. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 165–168. IEEE, 2017.

[24] Denae Ford, Justin Smith, Philip J Guo, and Chris Parnin. Paradise unplugged: Identifying barriers for female participation on stack overflow. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 846–857, 2016.

[25] Peter Freeman and David Hart. A science of design for software-intensive systems. *Commun. ACM*, 47(8):19–21, August 2004.

[26] Ryan J Gallagher, Kyle Reing, David Kale, and Greg Ver Steeg. Anchored correlation explanation: Topic modeling with minimal domain knowledge. *Transactions of the Association for Computational Linguistics*, 5:529–542, 2017.

[27] Erich Gamma, Richard Helm, Ralph Johnson, Ralph E Johnson, John Vlissides, et al. *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH, 1995.

[28] Barney G Glaser and Anselm L Strauss. *The discovery of grounded theory: Strategies for qualitative research*. Routledge, 2017.

[29] Nicolas E. Gold and Jens Krinke. Ethical mining. In *Proceedings of the 17th International Conference on Mining Software Repositories*. ACM, June 2020.

[30] Ian Gorton, John Klein, and Albert Nurgaliev. Architecture knowledge for evaluating scalable databases. In *2015 12th Working IEEE/IFIP Conference on Software Architecture*, pages 95–104, 2015.

[31] Hideaki Hata, Nicole Novielli, Sebastian Baltes, Raula Gaikovina Kula, and Christoph Treude. GitHub discussions: An exploratory study of early adoption. *Empirical Software Engineering*, 27(1), October 2021.

[32] Junxian He, Zhiting Hu, Taylor Berg-Kirkpatrick, Ying Huang, and Eric P Xing. Efficient correlated topic modeling with topic embedding. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 225–233, 2017.

[33] Abram Hindle, Neil A. Ernst, Michael W. Godfrey, and John Mylopoulos. Automated topic naming. *Empirical Software Engineering*, 18:1125–1155, 2012.

[34] Rashina Hoda. Socio-technical grounded theory for software engineering. *IEEE Transactions on Software Engineering*, 2021.

[35] Tuomo Korenius, Jorma Laurikkala, Kalervo Järvelin, and Martti Juhola. Stemming and lemmatization in the clustering of finnish text documents. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 625–633, 2004.

[36] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159, March 1977.

[37] Jintae Lee. Design rationale systems: Understanding the issues. *IEEE Expert: Intelligent Systems and Their Applications*, 12(3):78–85, May 1997.

[38] Diya Li, Harshita Chaudhary, and Zhe Zhang. Modeling spatiotemporal pattern of depressive symptoms caused by covid-19 using social media data mining. *International Journal of Environmental Research and Public Health*, 17(14), 2020.

[39] Hongwei Li, Zhenchang Xing, Xin Peng, and Wenyun Zhao. What help do developers seek, when and how? In *2013 20th Working Conference on Reverse Engineering (WCRE)*, pages 142–151, 2013.

[40] Georgios Liargkovas, Angeliki Papadopoulou, Zoe Kotti, and Diomidis Spinellis. Software engineering education knowledge versus industrial needs. *IEEE Transactions on Education*, pages 1–9, 2021.

[41] Tianyi Lin, Wentao Tian, Qiaozhu Mei, and Hong Cheng. The dual-sparse topic model: Mining focused topics and focused terms in short text. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, page 539–550, New York, NY, USA, 2014. Association for Computing Machinery.

[42] Victoria López, Alberto Fernández, and Francisco Herrera. On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed. *Information Sciences*, 257:1–13, 2014.

[43] Beth Lyall-Wilson, Nicolas Kim, and Elizabeth Hohman. Modeling human factors topics in aviation reports. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 63, pages 126–130. SAGE Publications Sage CA: Los Angeles, CA, 2019.

[44] A. MacLean, R. M. Young, and T. P. Moran. Design rationale: The argument behind the artifact. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 247–252. Association for Computing Machinery, 1989.

[45] Alvi Mahadi, Karan Tongay, and Neil A. Ernst. Cross-dataset design discussion mining. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 149–160, 2020.

[46] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. Design lessons from the fastest q&a site in the west. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, page 2857–2866, New York, NY, USA, 2011. Association for Computing Machinery.

[47] Saraj Singh Manes and Olga Baysal. How often and what stackoverflow posts do developers reference in their github projects? In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 235–239, 2019.

[48] Umme Ayda Mannan, Iftekhar Ahmed, Carlos Jensen, and Anita Sarma. On the relationship between design discussions and design quality: a case study of apache projects. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 543–555, 2020.

[49] Vukosi Marivate, Avashlin Moodley, and Athandiwe Saba. Extracting and categorising the reactions to covid-19 by the south african public - a social media study. In *2021 IEEE AFRICON*, pages 1–6, 2021.

[50] Matthew B Miles, Michae Huberman, and Johnny Saldana. *Qualitative data analysis: A methods sourcebook*. SAGE Publications, Incorporated, 2013.

[51] Saikat Mondal, C M Khaled Saifullah, Avijit Bhattacharjee, Mohammad Masudur Rahman, and Chanchal K. Roy. Early detection and guidelines to improve unanswered questions on stack overflow. In *14th Innovations in Software Engineering Conference (Formerly Known as India Software Engineering Conference)*, ISEC 2021, New York, NY, USA, 2021. Association for Computing Machinery.

[52] Tiago Oliveira Motta, Rodrigo Rocha Gomes e Souza, and Claudio Sant'Anna. Characterizing architectural information in commit messages: an exploratory study. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering*, pages 12–21, 2018.

[53] Gail C. Murphy, David Notkin, and Kevin J. Sullivan. Software reflexion models: Bridging the gap between design and implementation. *IEEE Transactions on Software Engineering*, 27(4):364–380, 2001.

[54] Leonardo Nizzoli, Serena Tardelli, Marco Avvenuti, Stefano Cresci, Maurizio Tesconi, and Emilio Ferrara. Charting the landscape of online cryptocurrency manipulation. *IEEE Access*, 8:113230–113245, 2020.

[55] Michael Nygard. Documenting architecture decisions, 2011.

[56] Rocco Oliveto, Malcom Gethers, Denys Poshyvanyk, and Andrea De Lucia. On the equivalence of information retrieval methods for automated traceability link recovery. In *2010 IEEE 18th International Conference on Program Comprehension*, pages 68–71. IEEE, 2010.

[57] Annibale Panichella, Bogdan Dit, Rocco Oliveto, Massimilano Di Penta, Denys Poshynanyk, and Andrea De Lucia. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In *2013 35th International conference on software engineering (ICSE)*, pages 522–531. IEEE, 2013.

[58] Yun Ning Pek and Kwan Hui Lim. Identifying and understanding business trends using topic models with word embedding. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 6177–6179. IEEE, 2019.

[59] Gustavo Pinto, Fernando Castor, and Yu David Liu. Mining questions about software energy consumption. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM Press, 2014.

[60] Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Michele Lanza. Mining stackoverflow to turn the ide into a self-confident programming prompter. In *Proceedings of the 11th working conference on mining software repositories*, pages 102–111, 2014.

[61] Paul Ralph and Yair Wand. A proposal for a formal definition of the design concept. In Kalle Lyytinen, Pericles Loucopoulos, John Mylopoulos, and Bill Robinson, editors, *Design Requirements Engineering: A Ten-Year Perspective*, pages 103–136, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[62] Kyle Reing, David C Kale, Greg Ver Steeg, and Aram Galstyan. Toward interpretable topic discovery via anchored correlation explanation. In *ICML Workshop on Human Interpretability in Machine Learning*, 2016.

[63] Martin P. Robillard. Sustainable software design. In *Proceedings of the International Symposium on Foundations of Software Engineering*, pages 920–923, 2016.

[64] Martin P Robillard, Andrian Marcus, Christoph Treude, Gabriele Bavota, Oscar Chaparro, Neil Ernst, Marco Aurélio Gerosa, Michael Godfrey, Michele Lanza,

Mario Linares-Vásquez, et al. On-demand developer documentation. In *IEEE International Conference on Software Maintenance and Evolution*, pages 479–483, 2017.

[65] Michael Röder, Andreas Both, and Alexander Hinneburg. Exploring the space of topic coherence measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, page 399–408, New York, NY, USA, 2015. Association for Computing Machinery.

[66] Sameh Nagui Saleh, Samuel A McDonald, Mujeeb A Basit, Sanat Kumar, Reuben J Arasaratnam, Trish M Perl, Christoph U Lehmann, and Richard J Medford. Public perception of covid-19 vaccines through analysis of twitter content and users. *medRxiv*, 2021.

[67] Abbas Shakiba, Robert Green, and Robert Dyer. Fourd: do developers discuss design? revisited. In *Proceedings of the 2nd International Workshop on Software Analytics*, pages 43–46, 2016.

[68] Camila Mariane C. Silva, Matthias Galster, and Fabian Gilson. Topic modeling in software engineering research. *Empir. Softw. Eng.*, 26(6):120, 2021.

[69] Mohamed Soliman, Matthias Galster, and Matthias Riebisch. Developing an ontology for architecture knowledge from developer communities. In *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, April 2017.

[70] Mohamed Soliman, Matthias Galster, Amr R. Salama, and Matthias Riebisch. Architectural knowledge for technology decisions in developer communities: An exploratory study with StackOverflow. In *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. IEEE, April 2016.

[71] Mohamed Soliman, Amr Rekaby Salama, Matthias Galster, Olaf Zimmermann, and Matthias Riebisch. Improving the search for architecture knowledge in online developer communities. In *2018 IEEE International Conference on Software Architecture (ICSA)*. IEEE, April 2018.

[72] M.-A. Storey, J. Ryall, J. Singer, D. Myers, Li-Te Cheng, and M. Muller. How software developers use tagging to support reminding and refinding. *IEEE Transactions on Software Engineering*, 35(4):470–483, July 2009.

[73] Xiaobing Sun, Xiangyue Liu, Bin Li, Yucong Duan, Hui Yang, and Jiajun Hu. Exploring topic models in software engineering data analysis: A survey. In *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 357–362, 2016.

[74] Fangchao Tian, Peng Liang, and Muhammad Ali Babar. How developers discuss architecture smells? an exploratory study on stack overflow. In *2019 IEEE International Conference on Software Architecture (ICSA)*. IEEE, March 2019.

[75] Jilles van Gurp and Jan Bosch. Design erosion: Problems and causes. *J. Syst. Softw.*, 61(2):105–119, March 2002.

[76] Bogdan Vasilescu. Academic papers using Stack Exchange data. `https://meta.stackexchange.com/questions/134495/academic-papers-using-stack-exchange-data`. Accessed: 2022-03-28.

[77] Greg Ver Steeg and Aram Galstyan. Discovering structure in high-dimensional data through correlation explanation. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[78] G. Viviani, M. Famelis, X. Xia, C. Janik-Jones, and G. C. Murphy. Locating latent design information in developer discussions: A study on pull requests. *IEEE Transactions on Software Engineering*, pages 1–1, 2019.

[79] Giovanni Viviani, Calahan Janik-Jones, Michalis Famelis, Xin Xia, and Gail Murphy. What design topics do developers discuss? In *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*, pages 328–3283. IEEE, 2018.

[80] Frank F Xu, Zhengbao Jiang, Pengcheng Yin, Bogdan Vasilescu, and Graham Neubig. Incorporating external knowledge through pre-training for natural language to code generation. In *Annual Meeting of the Association for Computational Linguistics*, ACL, pages 6045–6052. ACL, 2020.

[81] Di Yang, Pedro Martins, Vaibhav Saini, and Cristina Lopes. Stack overflow in github: Any snippets there? In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 280–290, 2017.

[82] Xin-Li Yang, David Lo, Xin Xia, Zhi-Yuan Wan, and Jian-Ling Sun. What security questions do developers ask? a large-scale study of stack overflow posts. *Journal of Computer Science and Technology*, 31(5):910–924, September 2016.

[83] Farida El Zanaty, Toshiki Hirao, Shane McIntosh, Akinori Ihara, and Kenichi Matsumoto. An empirical study of design discussions in code review. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–10, 2018.

[84] Haoxiang Zhang, Shaowei Wang, Tse-Hsun (Peter) Chen, and Ahmed E. Hassan. Are comments on stack overflow well organized for easy retrieval by developers? *ACM Trans. Softw. Eng. Methodol.*, 30(2), February 2021.

[85] Jie Zou, Ling Xu, Weikang Guo, Meng Yan, Dan Yang, and Xiaohong Zhang. Which non-functional requirements do developers focus on? an empirical study on stack overflow using topic analysis. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, May 2015.