Studying Trends, Topics, and Duplicate Questions on Q&A Websites for Game Developers

by

Arthur Veloso Kamienski

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering and Intelligent Systems

Department of Department of Electrical and Computer Engineering
University of Alberta

Abstract

The game development industry is growing and there is a high demand for developers that can produce high-quality games. These developers need resources to learn and improve the skills required to build those games in a reliable and easy manner. Question and Answer (Q&A) websites are learning resources that are commonly used by software developers to share knowledge and acquire the information they need. However, we still know little about how game developers use and interact with Q&A websites. In this thesis, we analyze the largest Q&A websites that discuss game development to understand how effective they are as learning resources and what can be improved to build a better Q&A community for their users.

In the first part of this thesis, we analyzed data collected from four Q&A websites, namely Unity Answers, the Unreal Engine 4 (UE4) AnswerHub, the Game Development Stack Exchange, and Stack Overflow, to assess their effectiveness in helping game developers. We also used the 347 responses collected from a survey we ran with game developers to gauge their perception of Q&A websites. We found that the studied websites are in decline, with their activity and effectiveness decreasing over the last few years and users having an overall negative view of the studied Q&A communities. We also characterized the topics discussed in those websites using a latent Dirichlet allocation (LDA) model, and analyze how those topics differ across websites. Finally, we give recommendations to guide developers to the websites that are most effective in answering the types of questions they have, which could help the websites in overcoming their decline.

In the second part of the thesis, we explored how we can further help Q&A web-

sites for game developers by automatically identifying duplicate questions. Duplicate questions have a negative impact on Q&A websites by overloading them with questions that have already been answered. Therefore, we analyzed the performance of seven unsupervised and pre-trained techniques on the task of detecting duplicate questions on Q&A websites for game developers. We achieved the highest performance when comparing all the text content of questions and their answers using a pre-trained technique based on MPNet. Furthermore, we could almost double the performance by combining all of the techniques into a single question similarity score using supervised models. Lastly, we show that the supervised models can be used on websites different from the ones they were trained on with little to no decrease in performance. Our findings can be used by Q&A websites and future researchers to build better systems for duplicate question detection, which can ultimately provide game developers with better Q&A communities.

Preface

The research work presented in this thesis has been conducted in the Analytics of Software, GAmes, And Repository Data (ASGAARD) lab led by Dr. Cor-Paul Bezemer.

Chapter 2 has been published as "A. Kamienski and C. Bezemer, 2021. An Empirical Study of Q&A Websites for Game Developers, Empirical Software Engineering Journal" [57]. The research conducted in this chapter received research ethics approval from the University of Alberta Research Ethics Board, Project Name "A study of Question and Answer websites for game developers", Project ID "Pro00101354", in June 12th, 2020. I was responsible for collecting and processing data from Q&A websites, building topic models, creating and sharing the survey with game developers, analyzing the data, and for manuscript composition. Dr. Bezemer was the supervisory author and was involved in concept formation and manuscript composition.

Chapter 3 has been submitted for review as "A. Kamienski, A. Hindle, and C. Bezemer, 2021. Analyzing techniques for duplicate question detection on Q&A websites for game developmers, Empirical Software Engineering Journal". I was responsible for collecting and processing data from Q&A websites, implementing question comparison techniques, analyzing results, and for manuscript composition. Dr. Bezemer and Dr. Hindle were the supervisory authors and were involved in concept formation and manuscript composition.

Acknowledgements

I would like to thank everyone that contributed in any way to the work presented in this thesis. First and foremost, I would like to express my gratitude to Dr. Cor-Paul Bezemer, who has provided invaluable advice and guidance throughout my studies. Dr. Bezemer has taught me much about conducting and presenting research in a reliable, responsible, and effective manner. I owe him this research project and the start of my research carreer.

I am also grateful to Dr. Abram Hindle for his guidance and feedback on my research work, and for being one of my thesis examiners. I would also like to thank Dr. Lei Ma for being my thesis examiner.

Thanks to all my friends that helped me during my studies, both in academic matters or otherwise. To all my friends at the ASGAARD lab, thank you for always being there whenever I needed support. A special thanks to Dener, who helped me set up and use several of the computational resources I used for my work.

Last but not least, I would like to thank my parents, my sister, and my wife for all of the love, care, and emotional support during my studies. You are my lifeline and never let me down, even in the toughest times.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and Background

1.1 Introduction

The game development industry has grown over the past few years [59] to become the largest in the entertainment segment with almost two hundred billion dollars of revenue in 2021 [129]. To keep up with the large growth, game development companies need to build high-quality games and require a skilled workforce of game developers. Meanwhile, game developers need ways to improve their skills and be on par with the best practices and latest technologies of the field.

Software developers in general face a similar situation and have shown a liking for Question and Answer (Q&A) websites to find the help and information they need [69]. The largest of those websites focused on software development is Stack Overflow[1], which currently receives millions of accesses per month [82]. While several researchers have studied Stack Overflow and how developers use it to share knowledge, there is still little information about how and if these Q&A websites help game developers. Despite game developers also using Stack Overflow, other websites such as Unity Answers[2], the Unreal Engine 4 (UE4) AnswerHub[3], and the Game Development Stack Exchange[4] have accrued hundreds of thousands of questions about game development and are valuable sources of data. Therefore, studying these websites can lead to

[1]https://stackoverflow.com/, accessed August 17, 2021.
[2]https://answers.unity.com/, accessed August 17, 2021.
[3]https://answers.unrealengine.com/, accessed August 17, 2021.
[4]https://gamedev.stackexchange.com/, acessed August 17, 2021.

valuable insights about how game developers use, interact and share knowledge in them.

In this thesis, we conducted two studies to analyze those Q&A websites from different perspectives and understand better their role as resources for game developers. In the first study, we explore the hypothesis that the Q&A websites are useful resources for learning game development skills. Meanwhile, the second study explores the hypothesis that we can help the websites by implementing better duplicate detection models with low resources. Based on our findings, we proposed measures that can help the studied websites in fostering better communities for their users, which can in turn increase their effectiveness in providing information to game developers. The two studies we conducted in this thesis were:

Research Study 1: An Empirical Study of Q&A Websites for Game Developers (Chapter 2)

Motivation: Despite several other studies having analyzed Q&A websites focused on discussing a variety of topics, no studies so far have analyzed Q&A websites for game developers. Therefore, in this study, we explore four Q&A websites that discuss game development and access their activity and effectiveness over time, the topics they discuss, and the perception of their users. By studying these websites, we seek to bring new understanding about how game developers use and interact in those websites, leading to insights into how we can help them become better resources for their users.

Findings: We found that the studied Q&A websites are in decline, with the number of posts and active users per month, the percentage of answered and resolved questions, and their answer effectiveness decreasing over the past few years. We also found that the release of new products is positively correlated to growth, while the percentage of answered and resolved questions are negatively correlated to it. Moreover, most of the topics discussed by the communities are specific to game development, and topics differ in many aspects such as the presence of code and the length of their

posts. Finally, users of Unity Answers and the UE4 AnswerHub who responded our survey had a negative view of those communities, preferred other online resources, and did not actively contribute to them.

Research Study 2: Analyzing Techniques for Duplicate Question Detection on Q&A Websites for Game Developers (Chapter 3)

Motivation: As we showed in Chapter 2, there are several challenges in building and maintaining an effective Q&A website. One of such challenges is dealing with duplicate questions, as they can hinder a website's effectiveness by overloading it with similar questions that may have already been answered, while making new questions which still lack answers harder to find. Currently, most Q&A websites deal with duplicate questions by offering users a feature to manually identify them. However, the high volume of duplicate questions makes the task of manual identification very burdensome to the websites' communities. In this study, we explore and compare several techniques for automatically detecting duplicate questions. Q&A websites can use our analysis of those different techniques for building better duplicate detection systems which can improve their ability of helping their users.

Findings: We found that comparing questions using the similarities between all their text content (i.e., title, body, and tags) along with their answers using a model based on MPNet offered the best performance for identifying duplicates. The similarity measures based on TF-IDF, BM25, and the Jaccard coefficient also showed good performance in ranking question pairs. We achieved higher performance when detecting duplicate questions using a supervised classifier trained on a small set of labelled questions pairs, almost doubling the results obtained by the best technique we analyzed. Finally, the supervised classifiers could identify duplicate questions on datasets other than the ones used for training them with only a slight decrease in performance.

While the findings from the first study provide insights and recommendations on how to help the studied Q&A communities and their users, the second study gives

a deeper understanding on how to improve the studied Q&A websites by providing them with better tools to increase their effectiveness. Together, the studies paint a thorough picture of Q&A websites for game developers and can be used by their communities and maintainers to build a thriving environment for game developers.

1.2 Q&A websites for game developers

Question and Answer (Q&A) websites allow users to interact with each other by posting and answering questions. Unlike other online forums where threads do not follow a specific format, Q&A websites offer a structure for organizing and separating posts into questions, comments, and answers. Threads are created when an user posts a question, and answers and comments can only be posted as a reply. While answers are meant to expose solutions to a question, comments are used for general discussions about a question or answer. Questions usually have tags attached to them to indicate the topic or broad category which they discuss, and can be marked as resolved when they receive an answer that correctly answers them.

There are several Q&A websites which discuss a variety of topics. While some websites do not focus on a single subject, others target specific niches such as science, business, and languages. For example, Quora[5] is a well-known Q&A website where users can ask question about any topic they like. Meanwhile, Stack Overflow[6], one of the most accessed websites on the internet, focuses on discussing questions about software, programming, and technology.

Some Q&A websites focus on game development topics, such as Unity Answers[7], the Unreal Engine 4 (UE4) AnswerHub[8], and the Game Development Stack Exchange[9]. Game developers also discuss game development issues on Stack Overflow, although in limited quantity. These websites host hundreds of thousands of posts

[5]https://www.quora.com/, accessed August 20, 2021.
[6]https://stackoverflow.com/, accessed August 20, 2021.
[7]https://answers.unity.com/, accessed August 20, 2021.
[8]https://answers.unrealengine.com/, accessed August 20, 2021.
[9]https://gamedev.stackexchange.com/, acessed August 20, 2021.

and are a valuable source of information about how game developers build games and interact with each other.

1.3 Thesis outline

The remainder of this thesis is organized as follows: Chapter 2 presents an empirical study on the trends, topics and user perception of four of the largest Q&A websites for game developers (Unity Answers, the Unreal Engine 4 (UE4) AnswerHub, the Game Development Stack Exchange, and Stack Overflow). Chapter 3 presents an analysis of the performance of different techniques for duplicate detection on the Game Development Stack Exchange and on Stack Overflow. Finally, Chapter 4 concludes the thesis by highlighting our findings and contributions and discussing possible future research directions.

Chapter 2

An Empirical Study of Q&A Websites for Game Developers

2.1 Abstract

The game development industry is growing, and training new developers in game development-specific abilities is essential to satisfying its need for skilled game developers. These developers require effective learning resources to acquire the information they need and improve their game development skills. Question and Answer (Q&A) websites stand out as some of the most used online learning resources in software development. Many studies have investigated how Q&A websites help software developers become more experienced. However, no studies have explored Q&A websites aimed at game development, and there is little information about how game developers use and interact with these websites. In this chapter, we study four Q&A communities by analyzing game development data we collected from their websites and the 347 responses received on a survey we ran with game developers. We observe that the communities have declined over the past few years and identify factors that correlate to these changes. Using a Latent Dirichlet Allocation (LDA) model, we characterize the topics discussed in the communities. We also analyze how topics differ across communities and identify the most discussed topics. Furthermore, we find that survey respondents have a mostly negative view of the communities and tended to stop using the websites once they became more experienced. Finally, we provide

recommendations on where game developers should post their questions, which can help mitigate the websites' declines and improve their effectiveness.

2.2 Introduction

The game development industry is growing [59] and hungers for experienced developers that can produce high-quality games [45]. This high demand for skilled game developers needs to be satisfied by the training of new developers in game development-specific abilities, and those seeking to develop games need ways to acquire the knowledge they need to hone their skills. Moreover, amateur and professional game developers alike need to stay updated on the most recent practices and technologies.

Developers in other fields of the software development industry face similar challenges and have shown a preference for question and answer (Q&A) websites (mostly Stack Overflow[1]) when it comes to acquiring new knowledge and seeking solutions to the problems they face [69]. Research has shown that these Q&A websites provide a fertile environment for knowledge sharing [8], and a large number of studies analyzed how these websites can help their communities by providing information and assistance [4]. Other studies have found similar results for general purpose [2], social [40], and health Q&A websites [49]. Yet, no research analyzed if and how Q&A websites related to game development help game developers.

Several Q&A websites specialize in game development, amassing hundreds of thousands of posts and offering a rich set of valuable data that could help to understand the game development environment. Studying these websites can provide us with a deeper understanding of the social and technical aspects of game development.

In this chapter, we explore the data that we collected from four Q&A websites, three of which target game developers (Unity Answers[2], the Unreal Engine 4 (UE4)

[1]https://stackoverflow.com/, accessed February 3, 2021.
[2]https://answers.unity.com/, accessed February 3, 2021.

AnswerHub[3], and the Game Development Stack Exchange[4]) and Stack Overflow, which targets all types of software developers. We analyze their content and activity to understand how game developers use them and what topics are discussed in them. We also use the 347 game developers' responses we received through a survey about two of these websites to analyze their users' behaviour and opinions. Specifically, we answer the following research questions (RQs):

RQ1. How did the studied game development Q&A communities evolve in terms of user participation?

We analyze the communities' activity and effectiveness over time to understand their ability of fostering an active user base. We also explore factors that are correlated to the changes we observed in the communities. We find that the communities have been in decline, with the number of posts and active users per month, the percentages of answered and resolved questions, and their answer effectiveness, measured as the percentage of answers marked as the accepted solution to their questions, decreasing over time. We also find that growth positively correlates to the release of new products, and negatively correlates to the percentage of answered and resolved questions. The most experienced users have a higher effectiveness and answered and solved a large number of questions, but the overall experience of the community decreased over time.

RQ2. What topics are most frequently discussed by game developers on the studied Q&A websites?

Game development requires specific software development knowledge and a diversified set of skills [58, 84]. However, there is still a lack of understanding of what those skills are, and how prominent they are to the game development community. With this in mind, this question explores the topics discussed by the Q&A websites in their questions, answers, and comments. We find that most of the topics we

[3]https://answers.unrealengine.com/, accessed February 3, 2021.
[4]https://gamedev.stackexchange.com/, accessed February 3, 2021.

identified are specific to game development, covering several subjects unique to it. Nevertheless, the most discussed topic in Unity Answers and the UE4 AnswerHub was *Bug reports*, while Stack Overflow and the Game Development Stack Exchange had many posts about object-oriented programming.

RQ3. What are the characteristics of posts from each topic?

Posts from game development Q&A websites about distinct topics may have different characteristics, be that in the way users write or interact with them. In this question, we explore if and how posts from the topics we identified differ in terms of specific aspects, such as the length of their text and the presence of code in them. We also compare posts belonging to a same topic across the studied Q&A websites to identify if there is any distinction in the way the communities discuss them. We show that topics were discussed with varying levels of abstraction and complexity, differing in terms of the presence of code and the length of their posts. Furthermore, we found that Unity Answers and Stack Overflow discussed code more frequently than the UE4 AnswerHub and the Game Development Stack Exchange.

RQ4. How do game developers perceive the studied communities?

We conducted a survey among game developers to provide a better view of how they regard Unity Answers and the UE4 AnswerHub and to complement our previous analyses. We chose to only ask questions about those two websites as their communities focus on their own game engine. We find that most respondents prefer other learning resources, such as online guides and tutorials, despite having accessed the Q&A websites in the past. Most respondents also do not actively contribute to the communities and even show disapproval for them. Moreover, respondents accessed the websites less frequently than they used to.

By answering these questions and bringing new knowledge about the studied game development Q&A communities, we shed light on how their users interact and discuss game development technologies. We also assess the overall health of the communities,

identify factors that relate to it, and characterize the topics in which they are the best and the worst at discussing. Our findings help game developers in understanding the characteristics of each of the studied Q&A websites, and we provide recommendations that can help them in choosing on which community to post their questions. By directing questions to communities that can handle them more effectively, we can help mitigate the communities' decline and increase their overall effectiveness.

The remainder of this chapter is organized as follows. In Section 2.3 we provide background information on Q&A websites and game engines, and discuss prior work conducted on these topics. Section 2.4 explains our methodology and Section 2.5 describes the motivation, approach, and findings for each of our RQs. We discuss the implications of our findings in Section 2.6, the threats to validity in Section 2.7, and conclude the chapter in Section 2.8.

2.3 Background and Related work

In this section, we provide background information on Q&A communities, Q&A websites for game development, and topic modeling of Q&A websites. We also provide an overview of other work related to this chapter and to these concepts.

2.3.1 Q&A communities

Question and Answer (Q&A) websites allow registered users to ask for help from peers by posing questions about the knowledge they seek and the problems they are facing. Users can interact either by asking, answering, or commenting on a question. Some Q&A websites, such as Yahoo! Answers[5] and Quora[6], cover a variety of subjects, while others specialize in specific areas like programming, cooking, photography, and academic research. These websites host large communities of users that share the same interest for a topic.

[5]https://answers.yahoo.com/, accessed February 5, 2021.
[6]https://www.quora.com/, accessed February 5, 2021.

Software developers are especially fond of Q&A websites, and a large number of these websites focus on different aspects of software development. The most prominent Q&A website aimed at developers is Stack Overflow, which hosts a large community of users and millions of questions about programming and technology. Stack Overflow currently receives over 120 million monthly visitors and ranks on the top 50 most popular websites in the world [82].

The large amounts of data generated through Q&A websites are an essential asset to understanding how the members of the communities behave, how they share knowledge among them, and how useful these websites are for users.

While there have been studies on other Q&A websites, software engineering researchers have invested enormous effort in studying Stack Overflow, and hundreds of papers have used its data since its initial release in 2008 [4]. Mamykina et al. [69] showed that Stack Overflow was more effective than other Q&A websites with high response rates and fast response times on the website's first years online while. Asaduzzaman et al. [9] explored why questions go unanswered and Wang et al. [124] identified factors that lead to fast answers.

Several studies have examined the topics users ask about [5, 6, 12, 14, 103, 125], their opinions [64], pain-points [29], and their overall asking and answering behaviour [106, 118]. Others have analyzed the quality of the knowledge that is shared. Zhang et al. [137] analyzed obsolete answers, while Parnin et al. [83] have compared the website's discussions to API documentation. Research has also shown the usefulness of the information shared on the website, exploring how developers use the code shared on the website [7, 131], and how to use it to augment other software development resources [36, 119].

Studies have also investigated methods of improving Stack Overflow, such as helping users find the information [77, 136], identifying expert users and using their knowledge to help the community [38], and increasing user contribution using symbolic rewards, such badges and bounties [20, 145].

Fewer studies explored the content of other Q&A communities. Hong et al. [49] have analyzed how users share knowledge on a health Q&A website, and Fu and Oh [40] analyzed the quality of answers on social Q&A websites. Other studies have provided ways of improving Q&A websites, such as increasing the overall politeness [126], increasing the popularity of academic answers [143], finding experts to answer questions [89, 110], predicting relevant discussions [90], and improving the way the community welcomes new users [108]. A significant portion of the research on other Q&A websites also focused on exploring what motivates users to participate in these communities [21, 31, 35, 41, 55, 142].

2.3.2 Game development Q&A websites

Among the many Q&A websites on the Internet, there are several that focus on discussing the development of games. Despite game development being composed of several skills ranging from programming to physics, these Q&A websites for game development usually revolve around a single game development engine and seek to help users find the solutions to the problems that surface when using that engine.

Game development engines are an integral part of the game development environment and shape the way games are made [39]. Unity and Unreal Engine 4 (UE4) are two of the most used game development engines, aggregating millions of registered users and being used to develop a large number of commercial games. These engines also ranked among the most popular technologies used by survey respondents on Stack Overflow's 2020 developer survey [81].

Both Unity and UE4 provide specialized Q&A websites, namely Unity Answers and the UE4 AnswerHub, to help engine users find the answers to their game development questions. Unity Answers and the UE4 AnswerHub are the two largest Q&A websites specializing in game development, hosting hundreds of thousands of questions about this topic. In comparison, Stack Overflow has less than sixty-five thousand questions related to these game engines, while the Game Development Stack Exchange hosts

fifty thousand questions in total, as of February 2021.

Game engines have seen various applications in research, such as simulating rockfalls [44], running behavioural experiments [18], and exploring urban areas [28]. There have also been studies on their performance [24, 71] and their use in serious game development [24, 26]. Studies on the game development community itself tend to focus on social aspects, such as gender [16, 17, 37] and the relationship between developers and game engines [127], rather than user behaviour and how they share knowledge. As far as we know, our study is the first to explore game development Q&A websites and their communities.

2.3.3 Topic modeling of Q&A websites

The Latent Dirichlet Allocation (LDA) model is a statistical generative model that identifies latent topics present in the set of documents given as input using a Dirichlet distribution [15]. The model assigns each word a probability of belonging to a given topic, therefore allocating them into groups. We can then use these probabilities to classify documents into the topics represented by the words. The LDA model is unsupervised, taking only the corpus and the number of topics to be identified as input, along with other tuning parameters. The final topics are unlabelled, and we need to identify their meaning (if any) by manually analyzing the words that compose them.

Many studies have used LDA for modeling topics from corpora of documents extracted from different sources [54]. The LDA model has also seen ample use in software engineering and mining software repositories research, where researchers used the extracted topics to analyze discussions related to different aspects of software development [22]. For example, Ray et al. [96] used the topics extracted from feature descriptions on GitHub to identify project domains, while Lukins et al. [68] used an LDA-based approach for automatic bug localization in open-source software. Other have used LDA to categorize and detect duplicate bug reports [19, 112], and analyze

reviews for mobile apps [51] and games [34].

Software engineering researchers have also amply used LDA to identify topics discussed on Stack Overflow. For example, Allamanis and Sutton [6] found that software developers discuss topics relating to how and why implementations and technologies do or do not work, while Barua et al. [12] analyzed how topics interact and change over time. Other studies analyzed the topics discussed on Stack Overflow about specific software development subjects such as machine learning [11, 42], mobile development [66, 103], and security [134]. Researchers have also analyzed the trends for topics of different software development disciplines finding that, for example, continuous engineering topics attracted fewer answers and are decreasing over time [135], while web development topics increased their share of questions [10]. We use similar techniques as those studies to analyze the topics of posts on Q&A websites for game development. Our study is, to the best of our knowledge, the first to analyze the topics discussed by game developers in Q&A websites.

2.4 Methodology

We used five distinct datasets from different sources for this study. These datasets comprise data we collected from four Q&A websites and from the responses game developers gave to our survey about these communities. In this section, we describe the process of collecting and processing this data before using it for our analysis. Figure 2.1 provides an overview of the steps taken in our methodology. The code and data used in this study can be found online in our replication package[7].

2.4.1 Data collection

We used data from four Q&A websites in this study, namely Unity Answers, the UE4 AnswerHub, Stack Overflow, and the Game Development Stack Exchange. We chose to analyse these websites as they are the four largest Q&A websites discussing game

[7]http://doi.org/10.5281/zenodo.5047790.

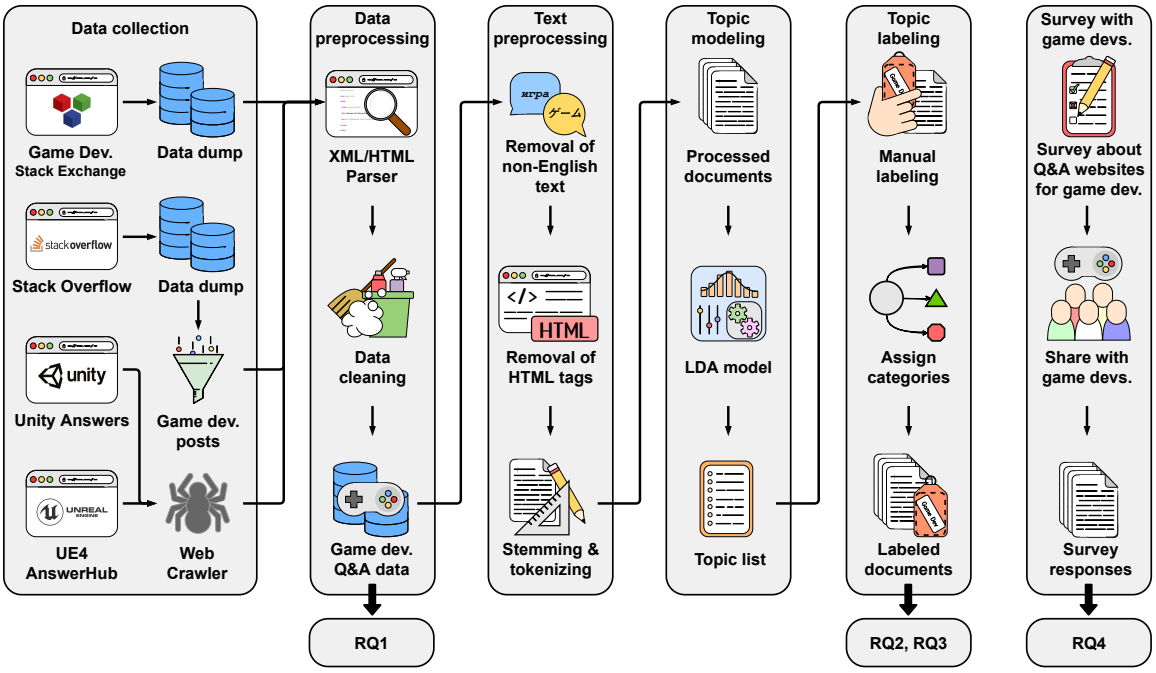


Figure 2.1: Overview of our data collection and processing methodology.

development that we could find.

We acquired the data from Stack Overflow and the Game Development Stack Exchange from the December 2020 Stack Exchange data dump[8]. For Stack Overflow, we selected posts (i.e., questions, answers, and comments) that had at least one of the following tags to compose our game development dataset: *'game-engine'*, *'game-physics'*, *'game-development'*, *'gameobject'*, *'2d-games'*, *'unreal-engine4'*, *'unreal-development-kit'*, *'unreal-blueprint'*, *'unrealscript'*, *'unityscript'*, *'unity-ui'*, *'unity-webgl'*, *'unity-networking'*, *'unity-editor'*, *'unity5'*, *'unity5.3'*, *'unity2d'*, *'unity3d'*, *'unity3d-mecanim'*, *'unity3d-5'*, *'unity3d-unet'*, *'unity3d-2dtools'*, *'unity3d-terrain'*, *'unity3d-gui'*, *'unity3d-editor'*. We manually selected these tags from the list returned by Stack Overflow's tag search engine[9] for the terms 'game', 'unity', and 'unreal'. We only selected tags related to overall game development and game engines and excluded tags related to other frameworks and programming languages, such as Python and JavaScript.

As opposed to the other websites, Unity Answers and the UE4 AnswerHub do

[8]https://archive.org/details/stackexchange, accessed February 5, 2021.
[9]https://stackoverflow.com/tags, accessed February 5, 2021.

not share their data in a structured format. Therefore, we created a web crawler to harvest data directly from the Unity Answers and UE4 AnswerHub websites. We used the websites' question catalogs, displayed on their homepage, as the starting point of crawling. These catalogs list the questions posted on the sites and contain IDs used to access them. We collected all of the question pages indexed by the catalogs, from oldest to newest. The pages also included the answers and comments posted as replies to the question. Likewise, we extracted the IDs of users that authored these posts and used them to access their profiles. We conducted the same procedure for Unity Answers and the UE4 AnswerHub. The crawler ran from August 3$^{\text{rd}}$ to August 11$^{\text{th}}$, 2020, collecting 593,006 pages from Unity Answers and 279,902 pages from the UE4 AnswerHub.

2.4.2 Data preprocessing

The data we collected is in XML format for Stack Overflow and the Game Development Stack Exchange, and HTML format for Unity Answers and UE4 AnswerHub. We therefore parsed those files into tabular format and selected the information we use in this study. For all posts, we selected the post ID, post date, author ID, and their text contents. We also selected the fields indicating to which post an answer or comment belongs, and the fields indicating if an answer was accepted and a question was resolved. For Unity Answers and the UE4 AnswerHub, we collected information about the type of user (e.g., moderator or staff member).

We performed several cleaning steps on the data, such as type conversions, date formatting, and removal of posts with missing information. Table 2.1 shows a summary of our data at the end of the cleaning process.

2.4.3 Text preprocessing

Prior to extracting topics from the posts for RQ2 and RQ3, we perform a series of text preprocessing steps. These steps aim to remove any of the text's unwanted features

Table 2.1: Summary of the data we collected from Q&A websites about game development.

Website	Questions	Answers	Comments	Posts'	Users
Unity Answers	381,055	347,264	652,091	1,380,410	196,954
UE4 AnswerHub	193,338	170,269	402,494	766,101	80,094
Stack Overflow	64,648	73,002	228,540	366,190	65,735
Game Dev. Stack Exchange	50,395	76,478	211,627	338,500	33,735
Total	689,436	667,013	1,494,752	2,851,201	376,481

'Posts = Questions + Comments + Answers

and provide better results in the next steps of the methodology.

We started by removing posts that belonged to any sections of the websites that were not primarily in English. We used a unique token to represent the presence of code in the texts, and replaced any code contained between the <code> HTML tag with it using regular expressions. Similarly, we replaced images using regular expressions that matched <img> tags, and URLs using regular expressions that matched <a> tags or a contiguous string of characters preceded by *http://* or *https://*. We removed any of the other HTML tags or symbols using Python 3's Beautiful Soup 4 library [99]. Finally, we used Python 3's Gensim library [97]'s `text_preprocess` function with default parameters to remove punctuations, multiple whitespaces, numeric characters, stopwords, and short words. The function also stems the texts using the Porter Stemmer [87], and tokenizes it by splitting words separated by spaces.

2.4.4 Topic modeling

We used two Latent Dirichlet Allocation (LDA) models to extract topics from the websites' posts. As Unity Answers and the UE4 AnswerHub are the official Q&A websites for their game engines, we believe the terminology used on them is different than that used on the other two websites. Therefore, we trained one of the models

using data from Unity Answers and the UE4 AnswerHub, and the other one using data from Stack Overflow and the Game Development Stack Exchange.

We used Gensim's implementation of the algorithm provided by its `ldamodel` function. We set the `alpha` and `eta` parameters to "auto", the `passes` parameter to 1, and the `eval_every` parameter to 5. We also used a value of 42 for the random state to ensure reproducibility. We defined the number of topics as 30, as we believed that was a good number of topics for an initial analysis. We experimented with other numbers of topics from 15 to 50, but found that the resulting topics were harder to interpret and did not provide satisfactory results.

We used the preprocessed texts obtained in Section 2.4.3 as input for the model. As an output, the model produced 30 sets of words accompanied by their probabilities of belonging to each set for each post. These sets represent the topics identified by the model based on the initial parameters. We used the word probabilities to assign the posts to each topic.

2.4.5 Topic labeling

The topics provided by the LDA models were initially unlabeled, given that they are only collections of co-occurring words that do not necessarily carry any meaning with them. Therefore, we manually labeled each of the topics from both models by analyzing their 15 most important words. We also randomly sampled and read ten posts from each topic to corroborate our manual classification. We excluded two topics from Stack Overflow and the Game Development Stack Overflow, as we could not find a suitable label for them.

After labeling the topics, we found that some of them comprised a large portion of the posts while providing little information regarding their content. Six of the topics (three from each model) were related to words commonly used in Q&A discussions, such as "help," "try," and "issue," and do not have a specific relation to game development. However, they are present in over 90% of posts and appear as the three

most important topics in most of them. We have thus opted to ignore these topics for our analysis by unassigning them to their posts. Table 2.2 and Table 2.3 show the lists of all of the topics, including their most important tokens.

We note that eliminating the topics left some posts without a topic assigned to them. The absence of a topic should indicate that the post discusses no other topic of interest. We opted to remove those posts from our analysis, as they only accounted for 0.6% (16,979) of the total. Thus, after removing the topics mentioned above, we obtained 27 topics for Unity Answers and the UE4 AnswerHub, and 25 topics for Stack Overflow and the Game Development Stack Exchange.

We further classified the topics into three broad categories to analyze the higher-level concepts discussed in the posts. We thus assigned any topic related to a specific programming language, paradigm, or concept to the "General software development" category and topics discussing errors, crashes, and bugs related to any tool, framework, platform, or programming language to the "Bugs, crashes and errors" category. We assigned the rest of the game development topics to the "Game development" category.

2.4.6 Survey with game developers

We ran a survey with game developers from June 16th to August 14th, 2020. We shared the survey on major online game development communities on websites such as Facebook and Reddit, and received 347 responses. A list of the surveyed communities is available online in our replication package.

We asked the developers multiple-choice and essay questions about their use of Unity, UE4, and their Q&A communities. The survey comprised 48 questions in 20 sections[10]. We directed respondents to different sections depending on their answers. For example, we asked questions about Unity only to those that used it for studying or working in the past.

[10]Survey available at https://figshare.com/s/a8fe3851e409967e2d39

Table 2.2: Summary of the labels and categories manually assigned to the topics obtained from running the LDA algorithm to data from Unity Answers and the UE4 Answerhub.

Category	**Topic**	**Top 5 tokens**
Game development	Game objects	object, compon, spawn, attach, destroi
	Meshes	mesh, node, map, scale, size
	Lighting	light, normal, shadow, graph, dynam
	Rendering	img, text, screenshot, imag, render
	Game mechanics	frame, block, speed, fp, damag
	Movement	rotat, move, direct, forward, movement
	Geometry	point, sphere, shot, cube, draw
	Game loop	time, run, second, continu, stop
	External tools	widget, plugin, thread, launcher, process
	Positioning	locat, pawn, cast, target, hit
	Collisions	collis, box, physic, overlap, wall
	3D modeling	import, model, templat, export, root
	Character animation	charact, plai, control, anim, state
	Materials	materi, textur, color, channel, bump
	Camera and display	camera, screen, world, main, view
	Game engines	project, engin, crash, build, file
	FX	sound, parent, child, item, hear
General software development	Runtime	save, content, load, game, devic
	General programming	privat, float, public, int, playercontrol
	Networking	player, server, client, connect, send
	GUI	click, select, button, input, press
	Object Oriented Programming	class, type, instanc, static, custom
	Event handling	event, tick, enabl, disabl, true
	File management	editor, file, asset, addit, data
Bugs, crashes and errors	Bug reports	issu, problem, fix, level, bug
	Programming errors	cpp, string, arm, format, lol
	General errors	error, compil, messag, fail, dll

Table 2.3: Summary of the labels and categories manually assigned to the topics obtained from running the LDA algorithm to data from Stack Overflow and the Game Development Stack Exchange.

Category	**Topic**	**Top 5 tokens**
Game development	Game objects	object, script, compon, variabl, gameobject
	Rendering	render, textur, shader, mesh, map
	Game environments	scene, agre, disabl, particl, enabl
	Viewport	window, platform, devic, opengl, mode
	Game mechanics	player, tile, enemi, spawn, mechan
	Movement	right, direct, turn, left, move
	Geometry	point, object, angl, distanc, box
	Positioning	posit, rotat, vector, transform, calcul
	Collisions	collid, physic, collis, forc, hit
	3D modeling	model, import, format, blender, score
	Character animation	anim, charact, item, grid, hand
	Display	size, scale, draw, target, unit
	Camera	camera, space, screen, world, coordin
	Game engines	uniti, project, error, build, version
	Sound/audio	sound, plai, step, sourc, cube
General software development	Runtime	data, save, load, text, store
	GUI	control, button, true, click, select
	General programming	valu, number, arrai, list, float
	User accounts	user, share, profil, pictur, cach
	Networking	server, client, send, connect, messag
	Publishing apps	support, app, googl, api, applic
	Java/Android development	main, thread, librari, coroutin, java
	Object Oriented Programming	type, class, instanc, entiti, static
	Event handling	input, state, event, action, inspector
	File management	imag, sprite, file, level, path

Table 2.4: Summary of the respondents of our survey.

	Category	**Responses**	
Age	≤17	20	(6%)
	18-29	223	(64%)
	30-39	80	(23%)
	≥40	23	(7%)
	Prefer not to say	1	(<1%)
Gender	Man	313	(90%)
	Woman	22	(6%)
	Other	5	(1%)
	Prefer not to say	7	(2%)
Game developer type (not mutually exclusive)	Hobbyist	141	(41%)
	Student	107	(31%)
	Professional	179	(52%)
Years of experience in game development	≤1	77	(22%)
	2-3	53	(15%)
	3-4	89	(26%)
	5-9	77	(22%)
	≥10	51	(15%)

All respondents also answered questions about their previous experience with game development and had the option to provide additional personal information such as their age and their country of residence. We used this information to characterize the population that originated our sample of respondents. Table 2.4 shows a summary of the responses we acquired for these questions. The majority of respondents are male game developers of less than 30 years of age. A significant portion of respondents (31%) are students, and many have only a few years of experience, which indicates that they are still early in their game development careers.

2.5 Results

In this section we discuss the motivation, approach and findings for each of our Research Questions.

2.5.1 RQ1. How did the studied game development Q&A communities evolve in terms of user participation?

Motivation: The goal of this research question is to uncover the trends experienced by the studied communities through a set of different measures. These trends provide insights on how successful the studied communities have been in nurturing and maintaining an active user base. We also identify a group of factors to investigate possible correlations to the changes we observed. Understanding what those factors are and how they relate to these trends allows us to derive actions for helping the communities prosper.

Approach: We used the creation dates of posts to analyze the evolution of the studied Q&A communities. We grouped the posts (questions, answers or comments) into months by removing the day from the post date. The monthly data provided less noise for temporal analysis. We removed the last incomplete month before the data collection, as it does not provide enough data for an accurate analysis. Overall, we grouped the posts into 131 months for Unity Answers (from October 2009 to July 2020), 78 months for the UE4 AnswerHub (from March 2014 to July 2020), 146 months for Stack Overflow (from August 2008 to December 2020), and 128 months for the Game Development Stack Exchange (from December 2009 to December 2020).

We used the number of total posts per month and the number of active users per month to measure community activity. We defined the number of active users as the number of unique users that created a post in a given month.

We analyzed the percentages of answered and resolved questions per month as a proxy for the communities' ability to help users who post questions. Answered questions are questions with one or more answers, while resolved questions are questions that have an accepted answer, indicating that the issue at hand was actually solved. We calculated these percentages by dividing the number of answered and resolved questions by the total number of posted questions in a month. Likewise, we measured the answer effectiveness per month, which we defined as the percentage of

accepted answers from the total number of answers posted in a month.

We used the Cox-Stuart test [27] to obtain the statistical significance of the trends in these measures. With this test, we test the null hypothesis of randomness against the alternative hypothesis of a decreasing or an increasing trend in the data. The Cox-Stuart test divides the sequence of observations in half and compares them by performing a sign test. The test yields a p-value (p) that we used to discard the null hypothesis if less or equal to a threshold of 0.05.

We moved on to identify factors correlated to the trends observed using the steps above. We identified thirteen major releases related to the Unity and UE4 game engines that occurred during the studied period. We defined major releases as the release of new products or updates in which there was a change in the most significant digit of the version number of the game engines. For Unity, we also included the release of version 3.5, as it introduced several major new features and was described by their creators as "one of the biggest additions to Unity since its inception" [117] at the time of its release. Major releases are associated with significant changes in the engines, which research has shown correlates to increased activity [65]. In addition, changes in the licensing models (such as the free and public releases for Unity and UE4) and the release of asset stores have made game development with those engines more accessible to the public, which may also lead to increased activity. Table 2.5 shows the list of releases we identified.

We compared the number of posts and active users in Unity Answers and the UE4 AnswerHub 30 days before and after each release to measure the correlation between these releases and the communities' trends. We only considered these two communities for the release-level analysis as they are focused on discussing a single engine each. All releases were more than 60 days apart, and the selected periods did not overlap. We left out three releases that occurred on the first or last 30 days of the observed period and did not provide a large enough sample for this comparison. We measured the difference between the distributions of the two 30-day intervals using

the median daily number of posts and active users. We ran the Wilcoxon-signed-rank test [128] to decide whether these differences were statistically different. We used the p-value (p) provided by the test to discard the null hypothesis of the two samples belonging to the same distribution if below a threshold of 0.05. This method is similar to the one used by Linares-Vásquez et al. [65] to identify the correlations between API changes and discussions on Stack Overflow.

We also calculated Cliff's delta (d) to show the magnitude of this difference [25]. The delta describes the proportion of times that a median from the first period is higher than one from the second. We used the interpretation proposed by Romano et al. [102] to define the effect size as negligible ($|d| \leq 0.147$), small ($0.147 < |d| \leq 0.33$), medium ($0.33 < |d| \leq 0.474$), or large ($0.474 < |d| \leq 1$).

We calculated the Pearson correlation coefficient to analyze how user activity, as measured by the number of active users and posts per month, correlates to the changes in the other measures.

We used the total number of posts made by each user as a proxy for their experience in using the Q&A websites. Users start with zero experience and become more experienced as they interact with the community. Therefore, we calculated each user's experience in each month after their first post to understand how user experience evolved.

Finally, we defined experienced users as the top 1% of users with the largest number of posts in each community (1,875 users on Unity Answers and 802 users on the UE4 AnswerHub). Other studies also analyzed the contributions of the top 1% of the most experienced and active users on Stack Overflow [69, 76]. We measured the contributions of experienced users by the number of questions they answered and resolved and their answer effectiveness.

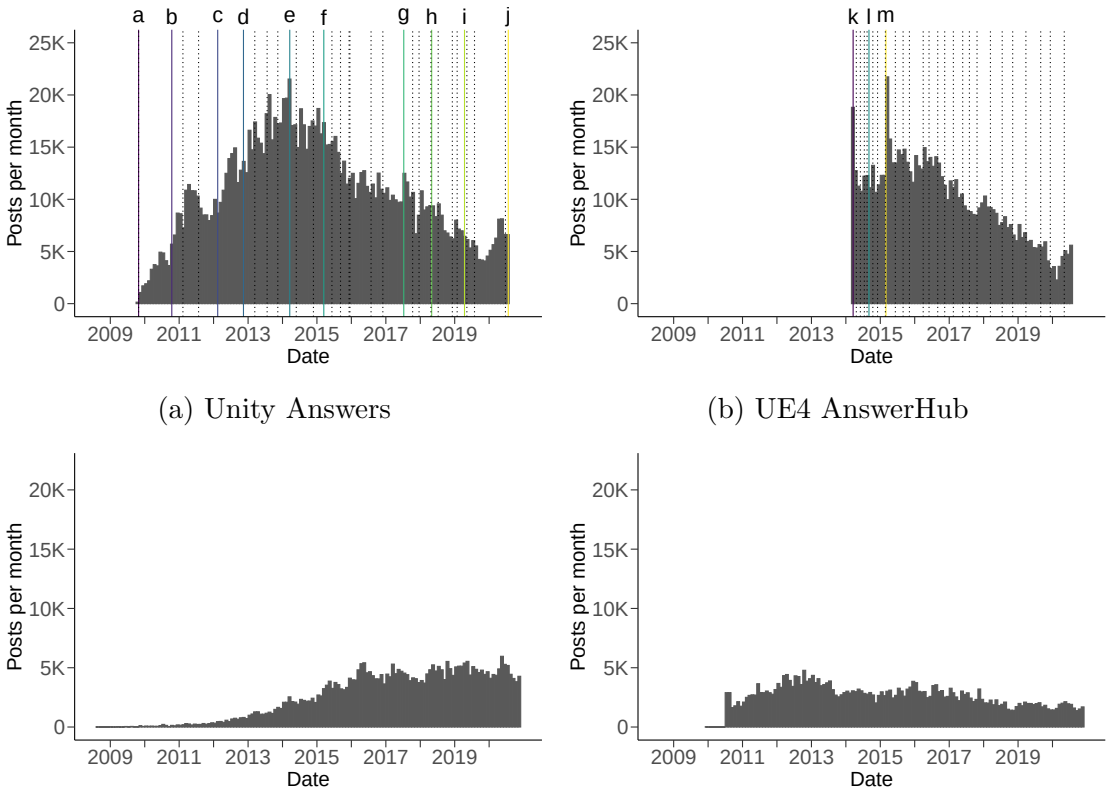
Findings: **Three of the studied communities have become less active over the past few years.** Figure 2.2 shows the number of posts per month for the four studied websites. We observed that the number of posts per month decreases on

the Game Development Stack Exchange after October 2012 ($p < .001$), on Unity Answers after March 2014 ($p < .001$), and on the UE4 AnswerHub after March 2015 ($p < .001$). These decreases occurred after periods of growth for Unity Answers ($p < .001$) and the Game Development Stack Exchange ($p = .03$), and stability for the UE4 AnswerHub ($p = .69$) after the communities' initial release.

We found an 81% decrease when comparing the maximum number of posts in March 2014 (21,525) to the lowest one in November 2019 (4,093) on Unity Answers, and a 90% decrease on the UE4 AnswerHub, with the number of posts going from 21,736 to 2,273 between March 2015 and February 2020. In the Game Development Stack Exchange the number of posts dropped from 4,785 in October 2012 to 1,380 in September 2020, representing a 71% decrease.

Table 2.5: Major releases for Unity and the UE4 game engines. We only study releases related to Unity Answers and the UE4 AnswerHub, as these websites focus on discussing a single game engine each.

Engine	**Label**	**Date**	**Description**
Unity	a	28-10-2009	Free release
	b	15-10-2010	v3.0
	c	14-02-2012	v3.5
	d	13-11-2012	v4.0
	e	19-03-2014	Asset store release
	f	15-03-2015	v5.0
	g	10-07-2017	v2017.1
	h	02-05-2018	v2018.1
	i	15-04-2019	v2019.1
	j	22-07-2020	v2020.1
Unreal Engine	k	19-03-2014	v4.0/Public release
	l	03-09-2014	Marketplace release
	m	02-03-2015	Free release



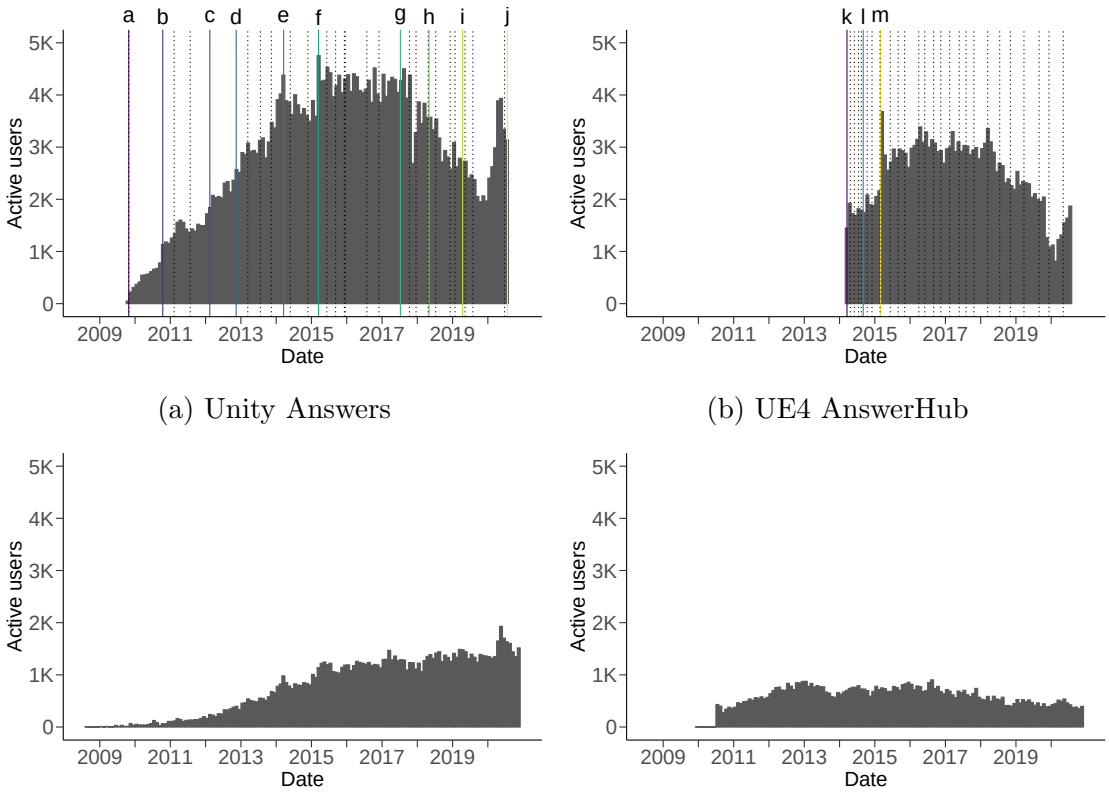
(a) Unity Answers

(b) UE4 AnswerHub

(c) Stack Overflow

(d) Game Development Stack Exchange

Figure 2.2: Evolution of the studied Q&A communities given by the number of posts per month. The solid vertical lines represent events that occurred throughout the communities' lifetime. The line labels (*a*-*m*) refer to the ones shown in Table 2.5. Dashed vertical lines represent minor updates.



(a) Unity Answers

(b) UE4 AnswerHub

(c) Stack Overflow

(d) Game Development Stack Exchange

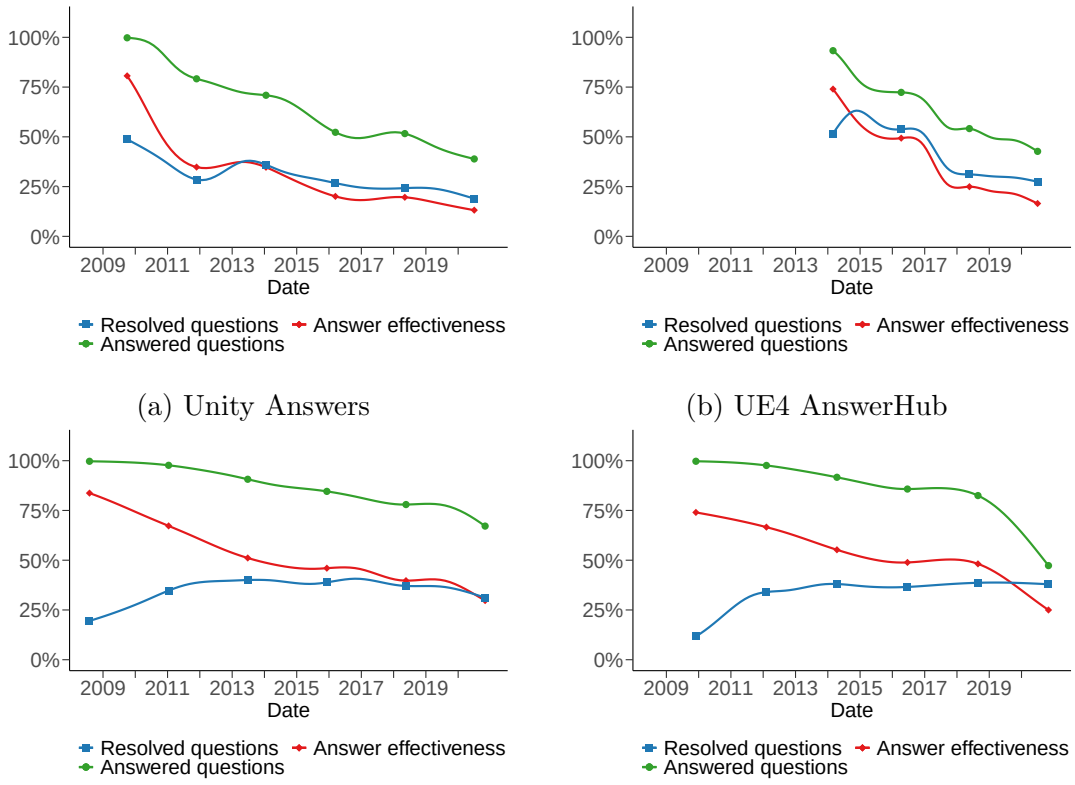
Figure 2.3: Evolution of the studied Q&A communities given by the number of active users per month. The solid vertical lines represent events that occurred throughout the communities' lifetime. The line labels (a-m) refer to the ones shown in Table 2.5. Dashed vertical lines represent minor updates.

We also observed the decline of those three communities (Unity Answers, the UE4 AnswerHub, and the Game Development Stack Exchange) in the number of active users per month (Figure 2.3). We noticed that the decline only started a few years after the decline in the number of posts per month in all of the communities. The three communities showed a similar pattern of early growth in number of active users followed by a period of stability before the decrease. On Unity Answers, the number of active users showed an initial growth until 2014 ($p < .001$) and a decreasing trend from late 2017 until early 2020 ($p < .001$). These trends are also present on the UE4 AnswerHub, where the number of active users grew until 2016 ($p < .001$) and started falling in 2018 ($p < .001$), and on the Game Development Stack Exchange, where that number grew until 2013 ($p < .001$) and decreased after 2016 ($p < .001$).

Between their highest and lowest points, the number of active users decreased 57% on Unity Answers, going from 4,481 to 1,953 between August 2017 and October 2019, 76% on the UE4 AnswerHub, going from 3,354 in March 2018 to 816 in February 2020, and 61% on the Game Development Stack Exchange, going from 904 in August 2016 to 351 in October 2020.

We noticed that these measures increased on Unity Answers and the UE4 AnswerHub during the first months of 2020, especially after March. These trends are likely a symptom of the COVID-19 pandemic, which may have increased the time game developers dedicated for learning and improving their skills. The number of posts and active users reached levels similar to what they were in 2018. On Unity Answers, the growth seems to be short-lived, with considerable declines occurring in June 2020. However, we note that these increases are still in development, and more observations are needed to analyze their overall trends.

In contrast to those three communities, we did not observe any decreases in posts or active users per month on Stack Overflow (Figure 2.2c and Figure 2.3c). Instead, the number of game development questions and active users discussing them grew until 2016 ($p < .001$), and plateaued after that ($p = .09$). The fact that Stack Overflow



(a) Unity Answers

(b) UE4 AnswerHub

(c) Stack Overflow

(d) Game Development Stack Exchange

Figure 2.4: Monthly percentages of answered and resolved questions, and answer effectiveness in the studied Q&A communities.

did not suffer a decrease as the other communities may be a consequence of it not being dedicated to discussing game development topics, and being a popular place for asking general software development questions.

The percentages of answered and resolved questions decreased over time. Figure 2.4 shows the percentages of answered and resolved questions for questions posted in each month in the studied communities. We found that in all of the communities, the percentages were at an all-time high during the first few months after their launch and decreased in most of the following months ($p < .001$ in all communities).

There were sharper declines on Unity Answers and the UE4 AnswerHub when compared to the other two websites. On Unity Answers, the percentage of answered questions remained above 97% during the first twelve months, and reached its lowest

point of 34% in June 2020. A similar decline occurred in the percentage of resolved questions, which peaked at 93% in October 2009 and reached a low of 11% in June 2020. On the UE4 AnswerHub, the percentage of answered questions fell from 97% in March 2014 to 37% in July 2020. Meanwhile, the percentage of resolved questions decreased from 81% in March 2014 to 14% in July 2020.

On the other hand, Stack Overflow and the Game Development Stack Exchange consistently kept the percentage of answered questions above 90% until 2013 and 2015, respectively. Furthermore, these percentages remained above 80% for most months until 2018, only showing steeper drops after that. Nevertheless, the percentage of answered questions reached 59% in November 2020 on Stack Overflow and 46% in April 2020 on the Game Development Stack Exchange. The percentages of resolved questions suffered similar decreases, going from 100% in October 2009 to 25% in November 2020 on Stack Overflow, and from 79% in September 2010 to 24% in September 2020 on the Game Development Stack Exchange. These percentages are higher than the ones found for all of the questions in Stack Overflow[11], where the percentages of answered and resolved questions peaked at 88% and 65% in 2009, and had a low of 45% and 24% in October, 2020.

Answers have become less effective on Unity Answers and the UE4 AnswerHub throughout the years. Figure 2.4a and Figure 2.4b show the answer effectiveness on Unity Answers and the UE4 AnswerHub. We found that the answer effectiveness reached its highest value in the first months, peaking at 53% in February 2010 on Unity Answers and at 67% in December 2014 on the UE4 AnswerHub.

The answer effectiveness has fallen on Unity Answers when analyzing the whole studied period ($p < .001$), despite experiencing an increase between February 2012 and September 2013 ($p < .001$). The lowest effectiveness we found on Unity Answers was 18% in June 2020. On the UE4 AnswerHub, the answer effectiveness increased

[11]https://data.stackexchange.com/stackoverflow/query/1284342/answered-and-resolved-questions#graph, accessed February 5, 2021.

from March 2014 to December 2014 ($p = .03$), but decreased over the following years ($p < .001$), reaching a low of 23% in June 2019.

We observed different patterns on Stack Overflow and the Game Development Stack Exchange. On those communities, the answer effectiveness grew until 2013 ($p = .005$ for Stack Overflow and $p < .001$ for the Game Development Stack Exchange) before stabilizing in the following years. Furthermore, on Stack Overflow the answer effectiveness started decreasing in 2017 ($p = .009$).

The communities grew as major releases occurred. We found that the release of Unreal Engine 4 for free in March 2015 correlates to an increase in the number of posts per day ($p < .001$, large effect size) and of active users per day ($p < .001$, large effect size) on the UE4 AnswerHub. When comparing the intervals prior and after the release, the median number of posts (456.5) and active users (212) per day increased over 50% (to 709 and 318.5, respectively).

On Unity Answers, a 35% increase in the median number of posts per day (from 146 to 197.5, $p = .003$, large effect size) and a 29% increase in the median number of active users per day (from 76.5 to 98.5, $p = .004$, large effect size) followed the release of Unity v3.0. The release of Unity v4.0 correlates to a smaller increase of 17% in the median number of active users per day (from 175.5 to 206, $p = .008$, large effect size).

These are only three of the ten major releases we analyzed. On Unity Answers, the release of the Unity Asset Store correlates to decreases of 19% (from 306.5 to 279.5, $p < .001$, large effect size) and 7% (from 728 to 592.5, $p = .02$, medium effect size) in the median number of posts per day and active users per day, respectively. Other major releases showed no statistically significant difference based on the test results. Nevertheless, the statistically significant releases correspond to the net growth in the communities. These results are consistent with the ones found for Stack Overflow, where changes in Android APIs correlate to increases in discussions [65].

The percentages of answered and resolved questions went down as the

communities grew. The increase in the number of posts and active users in all of the communities correlates to a decrease in the percentages of answered and resolved questions. We found correlation coefficients below -0.5 between these measures during the period of the communities' growth.

After the growth, we found correlations between 0.4 and 0.9 for all of the communities but Stack Overflow. For Stack Overflow, the correlations between the percentages and the number of posts remained at -0.5 during the period of stability after 2016. We found no correlation between the number of posts and the percentages of answered and resolved questions for Stack Overflow after 2016.

The decrease of the percentages while the community grew may be an indication of an overload caused by the large number of posts. On the other hand, the effects of a smaller and less active community may be playing a part in decreasing these percentages during the communities' decline.

Experienced users stopped contributing to the communities. Figure 2.5 shows the experience of active users over time on the studied communities. The overall experience of users started decreasing after 2013 on the Game Development Stack Exchange, after 2015 on Unity Answers and after 2017 on Stack Overflow and the UE4 AnswerHub. On the UE4 AnswerHub, 44% (34) of experienced moderators also became inactive in 2017.

This decrease occurred as experienced users become inactive, and new inexperienced users joined the community. While experienced users stopped using the communities, new users did not acquire enough experience to replace them.

Experienced users increased the percentage of answered and resolved questions. Experienced users significantly contributed to the communities by reducing the number of unanswered questions. We found that the top 1% most experienced users posted answers on 63% (149,794) of the 235,900 answered questions on Unity Answers, 67% (85,667) of the 128,100 answered questions on the UE4 AnswerHub, and 60% (26,157) of the 43,799 questions on the Game Development Stack Exchange.

Experienced users contributed slightly less on Stack Overflow, posting answers on only 43% (22,142) of the (51,971) answered questions.

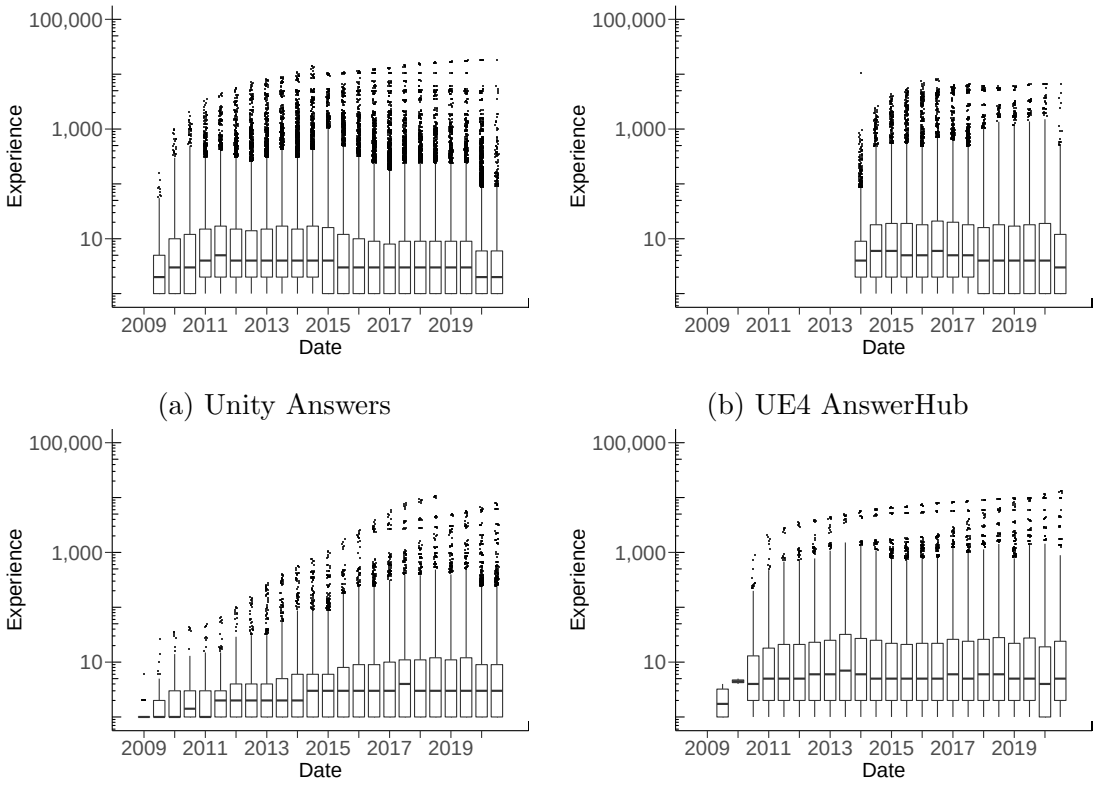
Experienced users also provided more effective answers. On Unity Answers, these users had an answer effectiveness of 39%, which is 18 percentage points higher than the effectiveness of other users (21%). On the UE4 AnswerHub experienced users had an answer effectiveness of 57%, compared to the effectiveness of 35% of others. This difference was smaller on the Game Development Stack Exchange, where experienced users had an effectiveness of 41% (against 30% of other users), and on Stack Overflow, where experienced users had an effectiveness of 48% (against 32% of other users).

Overall, experienced users resolved 66% (68,588) of the 103,723 resolved questions on Unity Answers, 68% (54,833) of the 80,749 resolved questions on the UE4 AnswerHub, 54% (14,484) of the 26,806 resolved questions on the Game Development Stack Exchange, and 43% (11,812) of the 27,460 on Stack Overflow.

We found that 10% (77) of the most experienced users we selected on the UE4 AnswerHub were moderators. These moderators had an even higher answer effectiveness (80%), answered 38,937 questions (30% of the total number of answered questions), and resolved 32,540 questions (40% of the total number of resolved questions). We also identified moderators among the most experienced users on Unity Answers, but their contribution to the community was far smaller than that of the UE4 AnswerHub's moderators.

Other studies have also found that a small percentage of experienced users contribute to Q&A communities by providing a large number of higher-quality answers [69, 76].

Summary: All of the studied communities have stopped growing during the last few years. The communities have become less active, and the percentage of unanswered questions has increased. Additionally, answers have become less effective, and most questions go unresolved. These changes correlate to the release of new products and engine versions, and to the decrease of the overall experience of the communities.



(a) Unity Answers

(b) UE4 AnswerHub

(c) Stack Overflow

(d) Game Development Stack Exchange

Figure 2.5: Distribution of active user experience per semester (measured in the cumulative number of posts made by each user) for the studied Q&A communities.

2.5.2 RQ2. What topics are most frequently discussed by game developers on the studied Q&A websites?

Motivation: Developing games requires a set of skills that are unique to the game development context [58, 84]. For example, other than the programming concepts needed to create and manipulate game behaviour, game developers also need a basic understanding of physics, graphic design, and sound editing. In this research question, we explore the topics of posts collected from the studied game development Q&A websites to identify how prominent they are in the community and how they have changed over time. This analysis reveals the topics in which game developers are most or least interested or those with which they require more or less help in understanding. Game development websites and their communities can use our findings to identify which topics they must focus on to help game developers in their work and improve their experience.

Approach: We analyzed the distribution of posts per topic and category in each of the studied communities. We identified the all-time most discussed topics and categories according to the proportion of posts assigned to them. We chose to analyze the proportion instead of the absolute number of posts as we are interested in exploring how these topics relate to each other. Furthermore, by using the proportion of posts, we avoid the effect of the growth or decline of the game development community.

We compared the distributions of posts belonging to each topic in the studied websites. As we obtained the topics from two different LDA models (see Section 2.4.4), we compared Unity Answers and the UE4 AnswerHub separately from Stack Overflow and the Game Development Stack Exchange. We used the Chi-square test to determine if there is a statistical significant difference between the distribution of posts per topic for each pair of websites. This test compares the frequencies of posts per topic between the websites to determine if they come from the same distribution. We rejected the null hypothesis of the distributions being the same if the resulting p-value is below the threshold of 0.05, indicating that the websites discussed topics

in different proportions.

While analyzing the topics in an all-time manner is useful for understanding historical and consolidated patterns, this approach hides possible trends that might have occurred over the years. The proportion of topics might change as time passes, and a long history of similar patterns might overshadow those that occurred in brief intervals. Thus, we use the time of posting collected with each post to calculate the distribution of topics per month.

We use the Cox-Stuart test [27] to identify any trend in the percentage of posts per month for each topic. We test the null hypothesis of randomness in the data against the hypothesis of non-randomness in the form of a decreasing or increasing trend. We then reject the null hypothesis if the resulting p-value is below a threshold of 0.00096 (0.05/52) that we obtained using the Bonferroni correction for 52 comparisons (one for each topic in Table 2.2 and Table 2.3).

Findings: **Most of the topics discussed on the studied Q&A websites are specific to game development.** Table 2.6 shows the number of topics belonging to the categories we defined. We found that 17 topics (63%) on Unity Answers and the UE4 AnswerHub are specific to game development, while 7 (26%) relate to general software development, and 3 (26%) relate to bugs, crashes, and errors. We found similar results on Stack Overflow and the Game Development Stack Exchange, with 16 topics (64%) being specific to game development, and 9 (36%) relating to general software development.

Table 2.6 also shows the number of posts assigned to each of the categories. We observed that 27% of posts on Unity Answers and 31% of posts on the UE4 AnswerHub belong to *Bugs, crashes and errors*, despite that category having only three topics assigned to it. On the other hand, only 15% of posts on Unity Answers and 13% on the UE4 AnswerHub belong to the *General software development* category. We did not identify any topics and posts relating to bugs, crashes and errors on Stack Overflow and the Game Development Stack Exchange, probably because these topics

Table 2.6: Distribution of topics and posts per category.

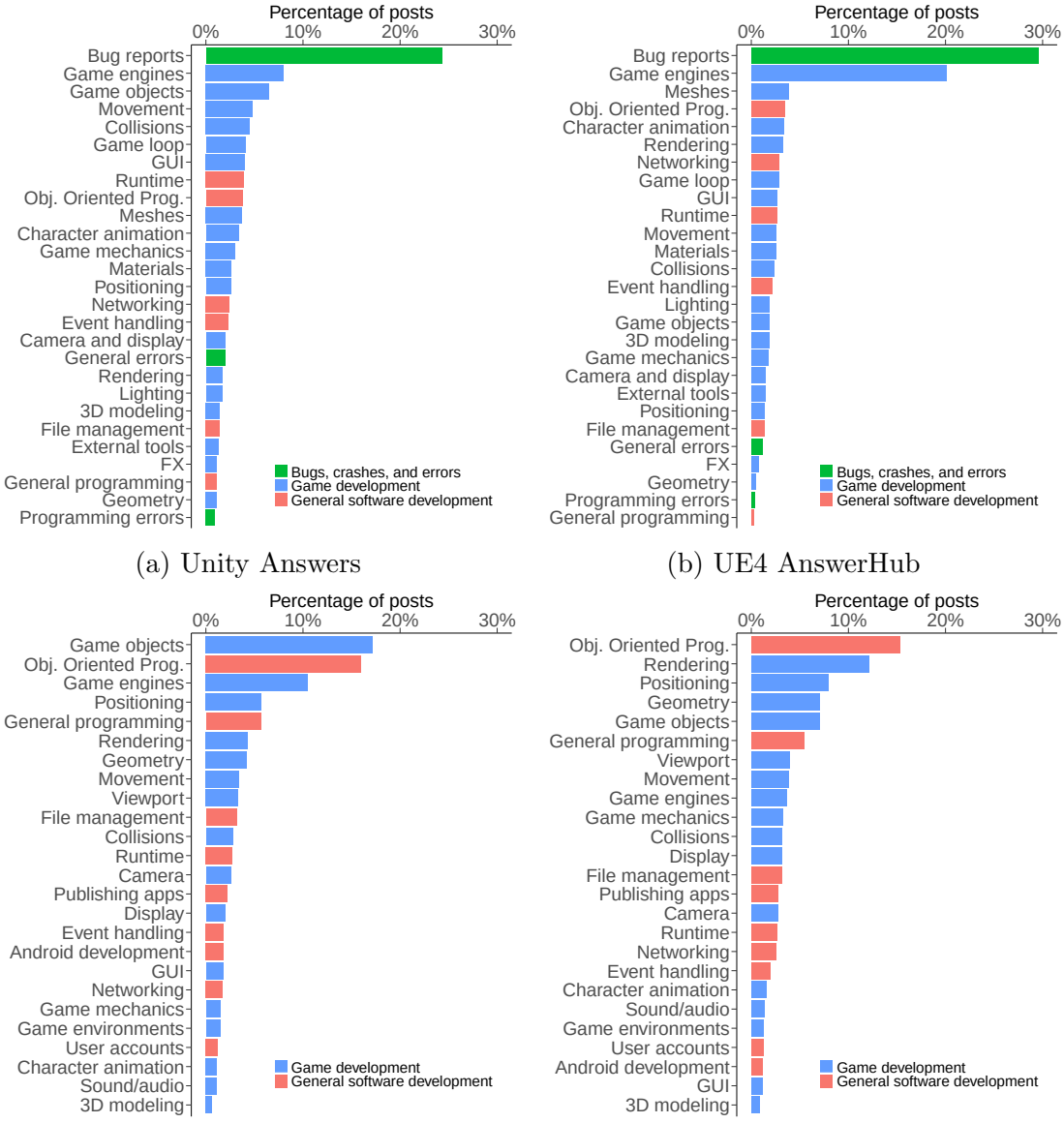
Website	Category	Topics	% of topics	Posts	% of posts
Unity Answers	Game development	17	63%	795,506	58%
	General software dev.	7	26%	207,053	15%
	Bugs, crashes and errors	3	11%	375,248	27%
UE4 AnswerHub	Game development	17	63%	423,439	56%
	General software dev.	7	26%	96,116	13%
	Bugs, crashes and errors	3	11%	233,383	31%
Stack Overflow	Game development	16	64%	232,702	64%
	General software dev.	9	36%	133,048	36%
Game Development Stack Exchange	Game development	16	64%	215,635	64%
	General software dev.	9	36%	122,092	36%

are specific to a game engine.

The distributions of topics show no statistically significant difference between the websites. Figure 2.6 shows the distribution of topics on the studied communities. We found a p-value of 0.24 after comparing both pairs of websites using the Chi-square test, and thus could not reject the null hypothesis of those samples coming from different distributions.

However, we still observed slight differences in the ranking of topics between the websites. For example, we found that *Game objects* was among the least discussed topics on the UE4 AnswerHub, while Unity Answers frequently discussed it, given that Game Objects are a specific type of object used as a basis for building characters, props and scenery in the Unity game engine. Similarly, Stack Overflow had a larger focus on discussing *Game objects* than the Game Development Stack Exchange. Those differences indicate that the websites have slightly different focuses, which may be a consequence of the different engines discussed on them and overall community preferences.

***Bug reports* was the most discussed topic on Unity Answers and the UE4 AnswerHub.** Figure 2.6a and Figure 2.6b show the percentage of posts assigned to



(a) Unity Answers

(b) UE4 AnswerHub

(c) Stack Overflow

(d) Game Development Stack Exchange

Figure 2.6: Percentage of posts assigned to each topic in the studied Q&A communities.

each topic on Unity Answers and the UE4 AnswerHub. We found that over 24% of posts on Unity Answers (334,839) and over 29% of posts on the UE4 AnswerHub (222,579) discuss *Bug reports*. Most of the other topics in those websites had fewer than 10% of posts assigned to them, with the only exception being *Game engines* on the UE4 AnswerHub. This finding indicates that game developers used Unity Answers and the UE4 AnswerHub as bug reporting platforms, as they are maintained by the companies that develop the game engines they discuss.

Game developers discussed different aspects of game development. Figure 2.6 shows the distribution of topics on the studied communities. We observed that topics related to game development are among the most discussed in all of the communities and describe many different aspects unique to game development. For example, topics such as *Meshes*, *Rendering*, and *Positioning* relate to game world-building, while *Movement*, *Collisions*, and *Character animation* discuss game behaviour. Other topics discuss the tools used for game development (e.g., *Game engines*), and the implementation of various game mechanics (e.g., *Game loop*).

Other game development topics have accrued a smaller number of posts and do not appear among the top-ranked ones. The lower-ranked topics may represent the ones in which game developers are less interested or have fewer issues. We also observed that some of these topics relate to specific niches inside the game development community, such as *3D modeling*, *Sound/audio*, and *FX*, and may therefore have a smaller group of developers discussing them.

***Object-oriented programming* was one of the most popular topics on Stack Overflow and the Game Development Stack Exchange.** Figure 2.6c and Figure 2.6d show the distribution of posts per topic on those two websites. We found that *Object-oriented programming* was the most discussed topic on the Game Development Stack Exchange with 15% of posts (51,671), and the second most discussed on Stack Overflow with 16% of posts (58,430). The *Game objects* topic was also popular on those websites, being the most frequently discussed on Stack Overflow

with 17% of posts (62,934), and the fifth most discussed on the Game Development Stack Exchange with 7% of posts (23,586). Game objects are the base class for entities in the Unity game engine, and game developers need to understand basic object oriented programming concepts to use them effectively.

Game developers also discussed *Object-oriented Programming* and *Game Objects* in the other websites, although in smaller proportions. Together, the two topics accounted for only 10% of posts on Unity Answers, and 5% on the UE4 AnswerHub.

The percentage of posts changed over time for several topics. Table 2.7 shows the number of topics with increasing and decreasing trends ($p < 0.00096$) for each of the studied communities. Over 30% of the topics showed changes in all of the communities, and over half on Unity Answers and the UE4 AnswerHub. We noticed that abrupt changes occurred at the start of the studied period, which can be explained by the low number of questions asked in the communities at that time. However, most of the topics had subtle changes of only a couple percentage points over the years.

Table 2.7: Summary of topic trends per website.

Website	Increasing topics	Decreasing topics	Topics changed	% of topics
Unity Answers	6	9	15	56%
UE4 AnswerHub	13	2	15	56%
Stack Overflow	8	2	10	40%
Game Development Stack Exchange	4	4	8	32%

Summary: Game developers discussed topics specific to game development more frequently than others. These topics covered several aspects unique to game development. The most discussed topic on Unity Answers and the UE4 AnswerHub was *Bug reports*, indicating that game developers used those websites as official bug reporting systems. Game developers also frequently discussed object-oriented programming, especially on Stack Overflow and the Game Development Stack Exchange.

2.5.3 RQ3. What are the characteristics of posts from each topic?

Motivation: In RQ2 (Section 2.5.2) we explored and compared the topics discussed in the studied game development Q&A communities using the distribution of posts per topic and category. While that analysis is important for identifying relevant topics and trends in the game development community, we still lack understanding of how these topics differ in practice. For example, game developers might discuss some topics frequently but in little detail, which may be of little use for those seeking to learn more about the subject. On the other hand, unpopular topics might offer more in-depth discussions, which provide a more reliable source of information. Therefore, analyzing the characteristics of the posts from each topic and how they differ might uncover the topics at which the game development community is proficient and those it should focus on improving, while also setting expectations for newcomers.

Exploring how topics differ among themselves is useful for understanding the game development community as a whole. However, we have collected texts from posts of four distinct Q&A websites which have different features, communities, and focus on specific aspects of game development. Therefore, we also explore how those topics differ in terms of the websites where they are discussed. A better understanding of the websites' specific characteristics can aid game developers in choosing with which community they will interact and be a part of based on their preferences.

Approach: We compared posts from each topic in each community based on six different aspects. We compared questions, answers and comments separately, as each has unique characteristics ensuing from their distinct uses in the websites. Table 2.8 describes the 10 comparisons we made (5 for questions, 3 for answers, and 2 for comments) based on the type of the posts, along with brief descriptions for each of them. The aspects we chose for these comparisons serve as proxies for measuring the quality of discussions for each topic. For example, the number of responses and answered questions point to how much help a user gets when asking a question.

Meanwhile, the number of resolved questions and accepted answers and the length of post texts gauge the quality of the help provided by other users by measuring if correct solutions are provided and how thoroughly they are discussed. Finally, the presence of code in posts indicates if the topics are discussed at a more technical or conceptual level.

Table 2.8: Description of the comparisons made for each topic based on the type of post being compared.

Post aspect	**Post types**	**Description**
Number of responses	Questions	The number of responses (comments or answers) added to a question
Resolved questions	Questions	Whether the questions received an answer marked as accepted
Answered questions	Questions	Whether the question received at least one answer
Accepted answers	Answers	Whether the answer is the correct solution to a question
Length of text	Questions, Answers, Comments	The length of the text contained in the post in characters (excluding code)
Presence of code	Questions, Answers, Comments	Whether the post contains a code snippet

We chose to compare one topic against the rest as opposed to all-vs-all, as we sought to identify the ones that stand out from the norm. The comparison between two topics may be relevant when validating a specific hypothesis about them, but its meaning becomes harder to interpret when analyzed as one of many other comparisons. We compare topics separately for each of the four studied websites, to avoid any bias caused by the differences between the communities. We also compared the high-level categories we identified in Section 2.4.5, but decided to leave them out of our analysis as they mirrored the results we found for individual topics. Therefore, we perform 1040 comparisons (27 topics $\times$ 10 comparisons, for Unity Answers and the

UE4 AnswerHub, and 25 topics $\times$ 10 comparisons for Stack Overflow and the Game Development Stack Exchange).

We used the Mann-Whitney U-test [70] to identify statistical significant differences between the distributions used in the comparisons. In this test, we reject the null hypothesis of two independent samples belonging to the same population if the resulting p-value is below a predefined threshold. To counteract the effect of multiple comparisons, we used the Bonferroni correction for 1040 comparisons, obtaining a threshold for the p-value of 4.8×10^{-5}.

Furthermore, we calculated the effect size of the statistical significant differences using Cliff's delta (d). We used the same thresholds as in RQ1 (Section 2.5.1) to determine if the difference was negligible ($|d| \leq 0.147$), small ($0.147 < |d| \leq 0.33$), medium ($0.33 < |d| \leq 0.474$), or large ($0.474 < |d| \leq 1$), following the interpretation proposed by Romano et al. [102].

We also compared the values of each aspect to provide a more palpable comparison. For aspects that represent binary values, such as resolved questions and accepted answers, we calculated the percentage of posts showing a positive value for that aspect. We compared other aspects using their median values.

We also used the aspects described in Table 2.8 to compare each topic across different websites. However, as opposed to the one-vs-rest comparison used previously, we compared the distribution of a topic in one of the websites with the distribution of the same topic on the other. We still performed 520 comparisons, resulting from the comparison of the distribution of 10 post aspects for each of the 27 topics in Unity Answers and the UE4 AnwerHub, and 25 topics in Stack Overflow and the Game Development Stack Exchange.

Once again, we used the Mann-Whitney U-test [70] to determine the statistical significance of the differences, using the same threshold of 9.6×10^{-5} for its p-value obtained from the Bonferroni correction for 520 comparisons. We also calculated the effect size of the differences using Cliff's delta (d), and classified the results using the

aforementioned criteria. Lastly, we compared the values of the aspects using the same percentages and medians described before.

Results: **Most topics are significantly different. However, most differences are negligible.** Table 2.9 shows a summary of the comparisons we performed for each aspect and each topic in the studied websites. Of the 1040 comparisons we performed, 676 (65%) showed statistical significant differences, meaning that the topics are different in most of the aspects we elected for comparison. Yet, we found that only 90 (13%) of these differences were non-negligible, and only four showed effect sizes above small.

Similarly, we found that topics differed in many aspects when comparing them across websites, but most of the differences were negligible. Out of the 520 comparisons we made, 383 (74%) were statistically significant. However, only 76 (20%) of these showed non-negligible effect sizes, 24 of them showing effect sizes above small. Our following analysis focuses on the non-negligible differences between topics.

The frequency in which game developers discussed code varied for some topics. Table 2.10 shows the topics with statistically significant and non-negligible differences in terms of presence of code in their posts studied communities. Nine of the topics we analyzed had more code in their posts than others. Notably, game developers discussed code more often in topics such as *Game Objects*, *Object-oriented programming*, and *General programming*. We note that these three topics relate to software development and programming concepts, which explains their higher percentage of code.

Positioning, *Movement*, and *Game loop* were the only other topics specific to game development to show a difference in terms of the percentage of posts discussing code in them. The presence of code indicates that those topics are focused on discussing implementation issues, while the rest of the game development topics discuss those subjects more conceptually.

Many topics differ in the length of the text in all post types. Table 2.11

Table 2.9: Summary of the comparisons we performed for each of the studied websites.

Website	Comparisons	Stat. significant	%	Non-negligible	%
Unity Answers	270	234	87%	35	15%
UE4 AnswerHub	270	189	70%	27	14%
Stack Overflow	250	131	52%	14	11%
Game Development Stack Exchange	250	122	49%	14	11%
Total	1040	676	65%	90	13%

shows the topics with statistically significant and non-negligible differences in terms of the median number of characters in questions and answers in the studied communities. 63 (70%) of the 90 non-negligible differences we found among topics were in terms of the length of the text contained in their posts. 37 of the differences occurred in terms of comments, 23 in terms of answers, and 3 in terms of questions. Those differences occurred in 21 different topics, with 9 of the topics only showing differences for comments. All of the topics had longer post contents than the others.

Some topics were longer in multiple websites and post types, such as *Networking*, *Collisions*, *Object Oriented Programming*, and *Positioning*. We found the largest differences between topics on the Game Development Stack Exchange, where the median character length of answers about game mechanics and networking was 908.5 and 869, respectively, compared to the median of 556 of others. The length of the text of a post may reflect the complexity and richness of the description used in the discussion. The fact that game developers provide more detail when discussing *Networking* and *Collision* may increase the effectiveness of the discussion by providing better answers for questions [43]. On the other hand, shorter questions may have superior quality than others [95] and may receive more useful answers [75].

Posts on Unity Answers and Stack Overflow discussed code more frequently than on the UE4 AnswerHub and the Game Development Stack Exchange. We observed that posts in Unity Answers contained code more frequently than in the UE4 AnswerHub for 23 topics in 46 comparisons. The differences were

Table 2.10: Topics with statistically significant and non-negligible differences with others in terms of presence of code in posts in the studied communities.

Topic	**Post type**	**Website**	**Code (%)**	
			In topic	**In others**
Game objects	Questions	Unity Answers	56	38
		Stack Overflow	91	61
		Game Dev. Stack Exchange	86	41
	Answers	Unity Answers	49	34
		Stack Overflow	92	44
		Game Dev. Stack Exchange	89	41
General programming	Questions	Stack Overflow	83	66
		Unity Answers	84	39
	Answers	Unity Answers	62	35
	Comments	Unity Answers	33	10
Object Oriented Programming	Questions	Unity Answers	63	39
		UE4 AnswerHub	33	10
	Answers	Unity Answers	56	34
		UE4 AnswerHub	27	7
General errors	Questions	Unity Answers	63	39
		UE4 AnswerHub	39	10
	Answers	UE4 AnswerHub	23	7
Positioning	Questions	Unity Answers	59	39
		Game Dev. Stack Exchange	63	45
	Answers	Game Dev. Stack Exchange	56	38
Movement	Questions	Unity Answers	56	39
	Answers	Unity Answers	49	34
Game loop	Questions	Unity Answers	56	39
File management	Answers	UE4 AnswerHub	23	8
Programming errors	Answers	UE4 AnswerHub	30	8

Table 2.11: Topics with statistically significant and non-negligible differences with others in terms of the number of characters in their questions and answers in the studied communities.

Topic	**Post type**	**Website**	**Median length**	
			Topic	**Others**
Networking	Questions	Game Dev. Stack Exchange	828	636
		Stack Overflow	654	545
	Answers	Game Dev. Stack Exchange	869	558
		Stack Overflow	430	320
		UE4 AnswerHub	321	253
		Unity Answers	288	226
Object Oriented Programming	Questions	Game Dev. Stack Exchange	761	636
	Answers	UE4 AnswerHub	308	252
		Unity Answers	291	224
Collisions	Answers	Stack Overflow	397	319
		UE4 AnswerHub	330	253
		Unity Answers	310	222
Positioning	Answers	UE4 AnswerHub	315	254
		Unity Answers	282	225
Materials	Answers	UE4 AnswerHub	314	253
		Unity Answers	286	225
Game mechanics	Answers	Game Dev. Stack Exchange	908.5	556
		Unity Answers	291	225
Game objects	Answers	UE4 AnswerHub	305	253
		Unity Answers	290	221
Movement	Answers	UE4 AnswerHub	319	253
		Unity Answers	317	223
Geometry	Answers	Unity Answers	292	226
Lighting	Answers	UE4 AnswerHub	353	253
FX	Answers	UE4 AnswerHub	335	254
Character animation	Answers	UE4 AnswerHub	316	253

small for 27 comparisons, medium for 15 comparisons, and large for 4 comparisons. The largest differences between the websites occurred in the *General programming* topic, with 84% of questions and 62% of answers on Unity Answers containing code, against 61% and 43% on the UE4 AnswerHub. The fact that the UE4 AnswerHub discussed less code may be an effect of its use of the Blueprints Visual Scripting system, which replaces the need of writing raw code in many tasks. The Unity engine did not have a visual scripting tool until August 2020.

We observed similar results when comparing Stack Overflow with the Game Development Stack Exchange. We found that 20 topics had more code on Stack Overflow, with small differences for 25 comparisons and medium differences for 5 comparisons. This time, the largest differences occurred in the *Game Mechanics* topic, with 76% of questions and 50% of answers containing code on Stack Overflow, against 37% and 17% on the Game Development Stack Exchange.

Summary: The topics showed significant differences in many aspects across the studied communities. Despite most of the differences being negligible, we could still identify topics that differed with at least small effect sizes in terms of the percentage code snippets in their posts and the length of their text. We also found significant non-negligible differences between posts belonging to the four websites, mainly in terms of the percentage of posts containing code.

2.5.4 RQ4. How do game developers perceive the studied communities?

Motivation: The data we collected from the studied Q&A communities allowed us to quantitatively study them through the interactions of users, and the tools provided by their websites. Our previous analyses are limited by the information we can infer from these interactions, and our previous findings build an incomplete image that may not reflect what developers undergo when using these communities. In this research question, we bridge the gap between our previous analysis and the actual way users experience the communities.

Approach: We used the 347 responses we collected from our survey with game

developers (Section 2.4.6) to study their perception of Unity Answers and the UE4 AnswerHub. We chose to ask questions only about these two websites, as they each have a unique community associated with their game engine. Only survey respondents who reported having used Unity or UE4 for studying or working answered questions about that engine's community. We further divided that group of respondents to isolate the ones that had actually used the studied Q&A websites.

We analyzed the frequency with which engine users accessed the Q&A websites. We split respondents into three mutually exclusive categories according to their access frequencies: those that had never accessed the site, those that had at least once, and those that did it regularly. We defined regularly as one or more accesses per month. Figure 2.7 represents the funnel that respondents went through when answering the survey, which allowed us to separate them into these groups.

Respondents who had accessed at least one of the Q&A websites indicated if they used the communities to ask questions, answer questions, report bugs, or search for a question related to theirs. We gave these respondents the option to write what they liked and disliked about the communities.

We asked respondents who regularly accessed the Q&A websites what the highest frequency with which they accessed the websites was. We also asked these respondents to classify their access as daily, weekly, or monthly. If their access frequencies changed, respondents chose their current frequency among those same options, with the addition of "less than monthly" and "stopped accessing." Respondents who changed their access frequencies or did not regularly access the websites could provide a reason for doing so.

We compared the respondents' usage of the Q&A communities to their usage of a set of other learning resources. We identified twenty resources available for Unity and nineteen for UE4. We asked engine users to indicate the frequency with which they accessed each resource for finding solutions to their game development questions. We split respondents according to their access frequency of these resources using the

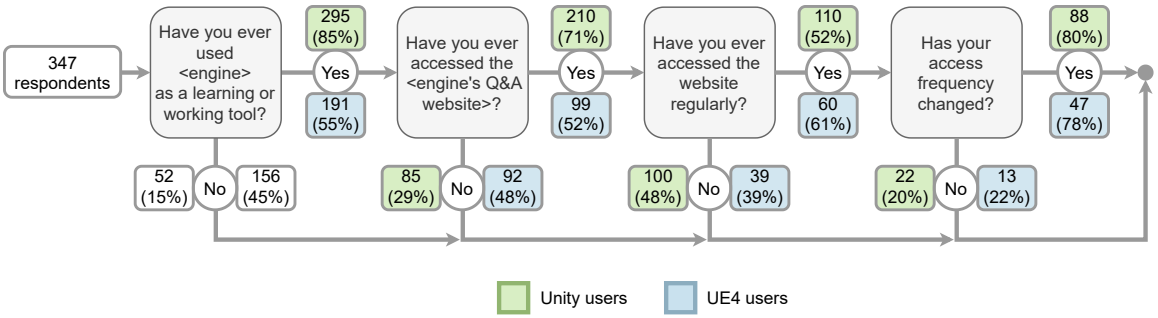
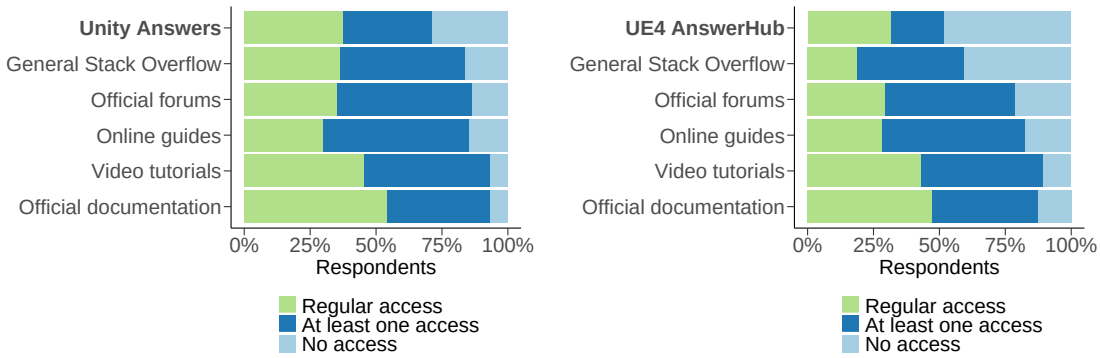


Figure 2.7: Representation of the selection process respondents went through when answering the questions of our survey. The boxes show the number of respondents which answered each of the questions with "Yes" or "No".

same categories we did for the Q&A websites.

Findings: **Respondents prefer other resources for finding solutions to their game development problems**. Figure 2.8a and Figure 2.8b show the access categories of respondents who used Unity and UE4 for the most frequently accessed learning resources, along with Unity Answers and the UE4 AnswerHub. We found that most respondents are aware of the existence of the studied Q&A communities. 71% (210) of 295 Unity users accessed Unity Answers, and 52% (99) of 191 UE4 users accessed the UE4 AnswerHub at least once in the past. These percentages are low when compared to the ones we found for other resources. A higher percentage of respondents reported having used the engines' documentation (93% of Unity users, 87% of UE4 users), video tutorials (93% and 89%), online guides (85% and 82%) and the engines' official forums (86% and 79%). A high percentage of Unity users (84%) also report having used Stack Overflow in general, likely looking for solutions related to the C# programming language used in the Unity engine. Considering the percentage of respondents that used each resource, Unity Answers ranked seventh out of the twenty resources we identified for Unity. The UE4 AnswerHub ranked eleventh out of the nineteen resources we identified for UE4.

In contrast, respondents accessed Unity Answers and the UE4 AnswerHub more frequently. 37% (110) of Unity users and 31% (60) of UE4 users reported having accessed the communities regularly. These percentages rank third when compared to



(a) Usage of learning resources for Unity (b) Usage of learning resources for UE4

Figure 2.8: Access categories of Unity and UE4 users for the most frequently accessed learning resources.

the other resources available for the engines.

Respondents who did not access the Q&A communities regularly also expressed their preference for other resources. When asked the reason for not accessing communities more frequently, 21 of the 49 respondents for Unity Answers and 5 of the 16 respondents for the UE4 AnswerHub mentioned using other means to find the solution to their problems.

Over half of the respondents did not actively participate in the Q&A communities. Only 40% (83) of respondents who accessed Unity Answers and 52% (51) who accessed the UE4 AnswerHub actively interacted with the communities by creating posts. Meanwhile, over 90% of respondents (191 on Unity Answers and 93 on the UE4 AnswerHub) used the communities to search for questions related to their problems.

Most of the respondents who reported having posted on the communities asked questions (71 on Unity Answers and 44 on the UE4 AnswerHub), while only a few gave answers (33 and 22). These results are similar to those found for Stack Overflow, where most users ask rather than answer questions [125]. On the UE4 AnswerHub, one in every five respondents (20%) also used the community to report bugs.

Respondents have a mostly negative view of the communities. When

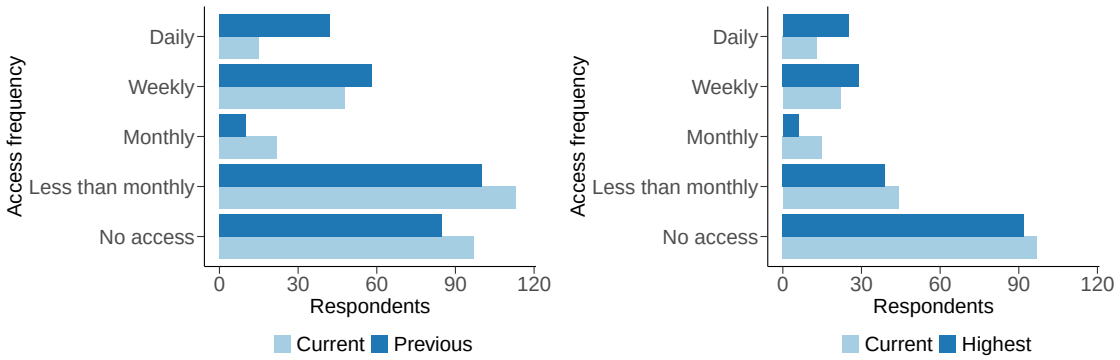
asked about their likes and dislikes about the communities, 52 (57%) of the 92 that answered the question expressed disapproval for Unity Answers and 23 (64%) of the 36 for the UE4 AnswerHub. In contrast, only 33 (36%) respondents indicated some approval for Unity Answers and 10 (28%) for the UE4 AnswerHub. Other responses were not related to the communities or did not convey a definite opinion.

For Unity Answers, 21 (23%) respondents complained about the low quality of answers. Eight (9%) of those respondents emphasized that many answers are obsolete and have broken links to other websites, which is also a problem faced by Stack Overflow [137]. Six (17%) respondents also mentioned the low quality of answers on the UE4 AnswerHub. Additionally, respondents criticized both communities for having unanswered questions (17 of the total 128 responses, 13%), for the overall website design and organization (seven responses, 5%), and the communities' low activity (11 responses, 9%). Six (5%) respondents mentioned the existence of resources other than Unity Answers for finding the answers they needed.

In contrast, thirteen of the 128 responses (10%) showed approval for the communities as they served their purpose of finding solutions. Other responses did not highlight the Q&A websites themselves but focused on the members of the community. Fifteen (12%) responses praised the communities for being large, supportive, active, and experienced.

The frequency with which respondents accessed the communities decreased. Figure 2.9a and Figure 2.9b show the distribution of the previous and current access frequencies reported by respondents who accessed the Q&A websites. We observed a decrease in the "daily" and "weekly" categories of access frequency in both communities, while the other categories increased. We found that over half of the respondents that accessed the communities regularly (65 of 110 for Unity Answers and 33 of 60 for the UE4 AnswerHub) either decreased or stopped their access to the communities.

The main reason respondents gave for decreasing their access was becoming more



(a) Respondents' access frequencies of Unity Answers

(b) Respondents' access frequencies of the UE4 AnswerHub

Figure 2.9: Current and previous frequencies with which survey respondents accessed the studied Q&A communities.

experienced and not needing the communities' help anymore. Some respondents also mentioned using other resources to find the answers they needed, while others reduced or stopped their use of the game engines. The respondents who stopped accessing the communities altogether reported similar motives as those that decreased their access. The only reason respondents gave for increasing their access frequency was the still existent necessity of answering their questions and acquiring more knowledge.

Summary: Despite knowing of the Q&A communities' existence, most survey respondents would rather use other learning resources. Of the ones that used the communities, most did not show support for them and did not post any questions or answers. The respondents who regularly accessed the communities decreased their access frequency as they acquired knowledge.

2.6 Implications of our findings

Our findings indicate that there are several challenges in maintaining healthy Q&A communities, especially when they become more popular and cater to a specific niche. All of the studied communities showed some sign of decline, despite those changes being more subtle in the smaller ones.

As the communities grew, new and inexperienced users joined and increased the number of questions. At the same time, experienced users that could provide solutions

to those questions became inactive for possibly feeling overloaded. More questions went unanswered and unresolved, causing frustration for newcomers. Users started seeking other resources for solving their problems, and community growth halted or even decreased. These communities could not yet recover after experiencing this decline.

Meanwhile, we have shown that the studied Q&A websites have different characteristics, discussing topics in different proportions and with varying levels of complexity of abstraction. These findings help in guiding game developers towards the websites that are most effective in answering the type of question they have. To further aid game developers in choosing the right Q&A website to post their questions, we created a flowchart (Figure 2.10) detailing our recommended decision process.

We based the flowchart on our analysis of the topics that each community is most effective in discussing. For example, we recommend that game developers post questions related to bugs and errors in Unity Answers and the UE4 AnswerHub given that those two websites had several posts related to these topics. Similarly, we recommend that game developers post programming questions on Stack Overflow or the Game Development Stack Exchange, as those topics were some of the most discussed in the two websites. We also direct game developers to the Game Development Stack Exchange if their questions fit a set of five topics (Rendering, Geometry, Movement, Game mechanics, and Positioning) that have been more effectively handled by that website.

By directing game development questions to the websites that are more likely to welcome and solve them, we can help the studied Q&A websites in mitigating their decline. For example, reducing the amount of questions asked in the largest websites and directing them to the smaller ones, we may avoid overloading a single community with a flood of questions. Moreover, we can increase the communities' effectiveness by reducing the number of questions that go unanswered and unresolved.

However, we note that many uncontrollable factors may be at play when it comes to

what is causing the communities' decline and Q&A website administrators should also act towards improving the websites and providing a better environment for nurturing a community. While we cannot infer causality and provide fail-proof advice, our findings also hint at actions that can help in lifting these communities, such as maintaining an active staff of moderators and investing in gamification features to motivate users' participation [8, 20, 31, 108, 126, 145].

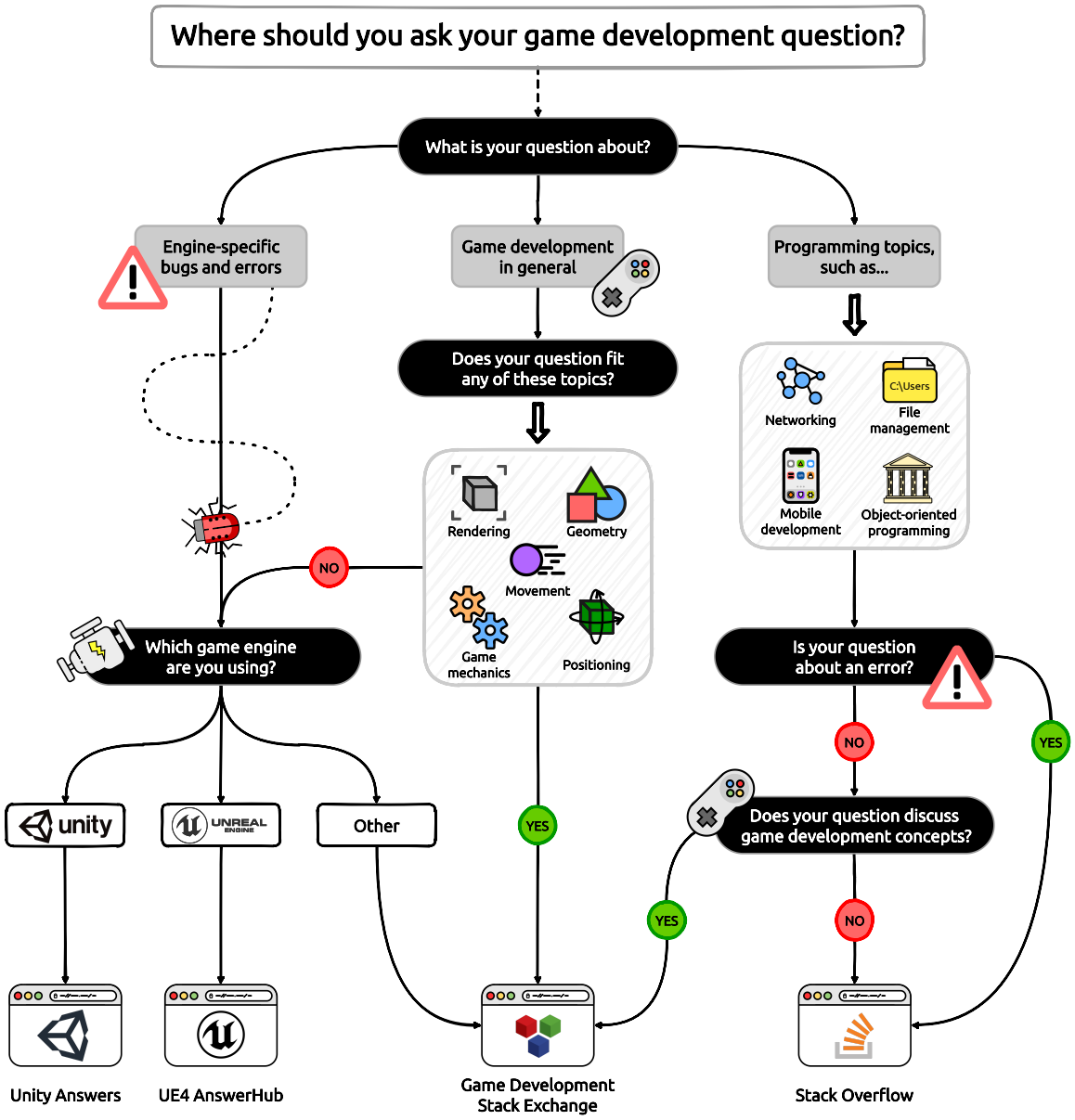


Figure 2.10: Flowchart showing our recommended decision process to choosing in which Q&A website to post questions about game development.

2.7 Threats to validity

In this section we discuss the threats to the internal and external validities of our study.

2.7.1 Construct validity

The choices we made in our methodology (Section 2.4) may have directly affected our findings. We have identified game development questions on Stack Overflow using a group of manually-selected tags. Therefore, we may have missed other game development questions with different tags than the ones we selected.

The LDA algorithm can provide different results based on its training input, and the preprocessing steps we perform on the data can have a direct impact in its output. Moreover, the output also depends on the initial parameters, and altering them might change the resulting set of topics.

We have experimented using additional preprocessing steps (such as pruning the corpora) and other sets of parameters for modeling topics. The results of these experiments were similar to the ones we presented, with slight variations in the meaning of topics and words assigned to them. Ultimately, we chose those that offered the best set of topics with our minimum possible interference.

Furthermore, we based our choice of the labels and categories assigned to each topic on our own experience and expertise with the game and software development disciplines. While we cannot guarantee that these labels precisely describe all of the subjects discussed in each topic, we bolstered our decision of labels with a manual analysis of the content of posts. Therefore, we are confident these labels can approximately describe their topics.

2.7.2 Internal validity

While we have not drawn any conclusions related to causality in our data, we have indicated measures that may help developers and communities based on our find-

ings (Section 2.6). However, we note that many unmeasured factors may have an effect on the patterns we observed. Thus, we cannot guarantee that the measures we recommended will have an effect on these websites and their communities. Yet, we still believe our findings are useful for game developers and game development communities by bringing more understanding of their past discussions.

The proposed taxonomy we used for classifying the topics obtained by the LDA algorithm was not validated by a video game expert. Future studies should validate the taxonomy in a large-scale study with a diverse group of game developers who have broad experiences with developing games with the studied game engines and some familiarity with the game development topics and discussions.

2.7.3 External validity

In this chapter, we have only analyzed data from three game development Q&A websites and questions related to game development on Stack Overflow. We note, however, that those websites are the largest of their kind, and we believe that they could reflect the trends in the game development community as a whole. Yet, our findings may be limited to the developer communities that interact with them. Future studies should investigate how our findings apply to other groups of developers that use other websites.

Furthermore, our findings may only apply to Q&A websites that discuss game development, and not to other types of websites or general-purpose Q&A websites. However, our analysis focuses on features common to many other Q&A websites and may serve as a foresight of what other communities experience if they meet similar conditions. Further studies should compare our findings to other Q&A communities.

Part of our study focused on analyzing specific aspects of the Unity engine and the Unreal Engine 4. While the majority of game developers use these two engines, future studies need to investigate how our findings hold for other game engines.

Our survey's responses come from game developers that were active members of

the selected game development communities when we shared the survey. Also, we only asked questions about Unity Answers and the UE4 AnswerHub, and our findings may not apply to other Q&A websites for game developers. Thus, this specific sample may not be an accurate representation of the whole game development community. Nevertheless, these respondents are still community members and their responses provide valuable information on the part of those communities they represent.

2.8 Conclusion

In this chapter, we analyzed game development Q&A communities using data from Unity Answers, the UE4 AnswerHub, the Game Development Stack Exchange, and game development-related questions from Stack Overflow. We also analyzed 347 responses game developers gave to our survey. We have explored how the communities evolved using measures of their activity and effectiveness, while also identifying what factors correlate to these measures' changes. We explored the topics discussed in those communities, analyzing these topics' distribution and identifying the most frequently discussed topics. We compared the topics among themselves and between the four websites using characteristics extracted from their posts. Finally, we examined how survey respondents used these communities through their access frequencies and their contributions to the websites. We also compared the respondents' usage of the websites to other online learning resources. Our most important findings are:

(1) The Q&A communities are declining, with their activity levels and effectiveness decreasing over time.

(2) Experienced users contribute to the community by answering and resolving a large number of questions. These users stopped using the community, reducing the communities' effectiveness.

(3) Game developers mainly discussed topics related to game development, with topics about Bug Reports having accrued a larger number of posts on Unity Answers and the UE4 AnswerHub.

(4) Topics have shown varying levels of abstraction and complexity, differing in terms of the length of their posts and the frequency with which code is discussed in them.

(5) Users do not often participate by posting questions or answers, have a mostly negative view of the community, and have reduced their access frequencies.

These findings bring new understanding about game developers, their communities, and the topics that they discuss. Based on these findings, we provide recommendations that help game developers in directing their questions to communities that can handle them better, which can aid the communities to overcome their decline and become more effective in the future.

Chapter 3

Analyzing Techniques for Duplicate Question Detection on Q&A Websites for Game Developers

3.1 Abstract

Game development is currently the largest industry in the entertainment segment and has a high demand for skilled game developers that can produce high-quality games. To satiate this demand, game developers need resources that can provide them with the knowledge they need to learn and improve their skills. Question and Answer (Q&A) websites are one of such resources that provide a valuable source of knowledge about game development practices. However, the presence of duplicate questions on Q&A websites hinders their ability to effectively provide information for their users. While several researchers created and analyzed techniques for duplicate question detection on websites such as Stack Overflow, so far no studies have explored how well those techniques work on Q&A websites for game development. With that in mind, in this chapter we analyze how we can use pre-trained and unsupervised techniques to detect duplicate questions on Q&A websites focused on game development using data extracted from the Game Development Stack Exchange and Stack Overflow. We also explore how we can leverage labelled data to improve the perfor-

mance of those techniques. The pre-trained technique based on MPNet achieved the highest results in identifying duplicate questions about game development, and we could achieve a better performance when combining multiple unsupervised techniques into a single supervised model. Furthermore, the supervised models could identify duplicate questions on websites different from those they were trained on with little to no decrease in performance. Our results lay the groundwork for building better duplicate question detection systems in game development Q&A websites and ultimately providing game developers with a more effective Q&A community.

3.2 Introduction

The video game industry is currently the largest entertainment industry in the world, having accumulated almost 180 billion dollars in revenue in 2020 and surpassing global movie and North American sports industries [129]. Behind all of the games the industry produces, there is a large number of game developers that help conceive, create, and build each of them. A strong growth is still expected for the future of the video game industry and the game development community needs to be prepared to supply skilled workers to satiate this demand.

Several online and offline resources seek to teach game development skills to aspiring developers, thereby being an important asset to train new developers. An example of such resources is Question and Answer (Q&A) websites, which are popular choices for knowledge sharing among game developers. Some of the largest Q&A websites for game development, such as Unity Answers[1], the Unreal Engine 4 (UE4) AnswerHub[2], and the Game Development Stack Exchange[3], host hundreds of thousands of discussions about many aspects of game development and are a valuable source of knowledge for those seeking to learn about those topics.

However, maintaining a healthy and active Q&A community that can effectively

[1]https://answers.unity.com, accessed September 6th, 2021.
[2]https://answers.unrealengine.com, accessed September 6th, 2021.
[3]https://gamedev.stackexchange.com, accessed September 6th, 2021.

help its members is a hard task and comes with several challenges. One of these challenges is dealing with the large number of duplicate questions posted by users [144], which can negatively impact the websites. For example, users who are willing to answers questions have the additional burden of manually filtering through and marking questions which are duplicate, while question askers may experience increased wait times to get their responses.

Duplicate questions can be even more hurtful for Q&A websites for game development. According to our prior work, the four largest websites that discuss game development topics have been in decline over the last few years [57], and the presence of many duplicate questions may further decrease their effectiveness. Additionally, detecting duplicate questions is a hard task for websites such as Unity Answers and the UE4 AnswerHub, as they do not provide users with a feature of manually tagging duplicate questions. As a consequence, these Q&A websites and their moderators have to dedicate many resources to reducing the effects of this phenomenon.

With that in mind, several researchers have proposed methods to automatically identify duplicate questions, which reduce the effort of manually identifying such questions and can help prevent the posting of new duplicates [3, 111, 121, 122, 138, 139, 144]. While researchers have put a lot of effort into identifying duplicates on large and popular Q&A websites such as Quora[4] and Stack Overflow, little is known about how duplicate question detection techniques adapt to the context of game development. Previous work has shown that the performance of duplicate detection techniques in Stack Overflow can vary according to the programming language being discussed [3, 121, 122, 138, 139]. Similarly, the performance of these techniques may be affected by the specific characteristics of different communities or different question topics such as game development.

Identifying duplicate questions on Q&A websites for game development is an arduous task, as sources of labelled data are scarce and developing custom-tailored

[4]https://www.quora.com, accessed September 6th, 2021.

techniques from scratch can be very computationally expensive. Furthermore, it is very hard to reuse previously proposed techniques developed for the software engineering domain (e.g., Stack Overflow), as almost no studies provide resources for implementing and reusing their proposed approaches.

Therefore, in this chapter, we analyze how existing pre-trained and unsupervised techniques can be used to detect duplicate game development questions. We also introduce new techniques which have not been previously used for the task of detecting duplicate questions in the software engineering domain. We evaluate the performance of those techniques using labelled game development data from the Stack Exchange data dump. We also use the labelled data to train supervised models and evaluate their performance at detecting duplicate questions in different datasets, including the ones that were not used for training them. More specifically, we explore the following research questions (RQs):

- **RQ1. What is the performance of unsupervised and pre-trained techniques for duplicate question detection on game development Q&A data?**

  There are several techniques for measuring the similarity between two documents that do not rely on a labelled set of data. These techniques are valuable for websites that do not offer a feature for tagging duplicate questions, thus not having a source of data for training supervised models. In this research question, we test and compare seven different techniques to evaluate how they perform on the task of identifying duplicate questions about game development. We find that computing the similarity between all question elements (i.e., title, body, tags, and answers) using a model based on MPNet [113] provides the best results for the task.

- **RQ2. How can we leverage labelled data to improve the performance of unsupervised techniques?**

  Despite being relatively small, the set of labelled duplicate questions we acquired from the studied websites can still prove useful for improving the results

we obtained in RQ1. Furthermore, the techniques we explored in RQ1 use different methods for characterizing duplicate questions, and aggregating them into a single metric may help us achieve even higher performance. In this question, we use the similarity scores obtained by the unsupervised techniques and the set of labelled data to build a supervised model to compare questions and provide a new similarity score. Using this model, we could almost double the recall-rate@k score of the best technique we found in RQ1 for game development questions on Stack Overflow. We also found that we can use the supervised models for classifying duplicate questions on different websites with little to no decrease in performance.

The answers to these questions can provide important knowledge regarding the task of identifying duplicate questions on Q&A websites. By analyzing how to use existing unsupervised and pre-trained techniques and how to leverage labelled data to aid in this task, we lay the groundwork for Q&A websites with low resources (such as those focused on game development) and future researchers to build systems that can detect duplicate questions more reliably. Our main contributions are:

- We explore and compare seven unsupervised and pre-trained techniques for duplicate question detection of ranging complexities, laying the groundwork for the development of unsupervised duplicate detection systems;
- We introduce and analyze two new techniques for detecting duplicate questions based on BERTOverflow [116] and MPNet [113], which have not been previously used in the software engineering domain;
- We show that using answers can improve the performance of the studied techniques;
- We show that a small set of labelled data can be used for improving the performance of duplicate detection systems;
- We show that supervised models can be used for detecting duplicate question

on websites other than the ones in which they were trained with little to no decrease in performance;

- We provide recommendations for developing systems for duplicate question detection, such as the best approaches for choosing candidate question pairs prior to training and evaluating supervised models, and outlining the common pitfalls that can occur when designing those systems;

- We provide a replication package containing all data and techniques used in this chapter, allowing researchers to use, reproduce, and evaluate our results in future studies.

The remainder of this chapter is organized as follows: Section 3.3 discusses background and related work. Section 3.4 describes our methodology and we present our findings in Section 3.5. In Section 3.6 we discuss the matter of comparing our methodology to those of other studies, and in Section 3.7 we discuss the implications of our findings. Section 3.8 presents the threats to the validity of our study. Finally, Section 3.9 concludes the chapter.

3.3 Background and related work

In this section we provide an overview of some of the concepts discussed in this chapter and of other related work.

3.3.1 Q&A websites

Question and Answer (Q&A) websites are places of knowledge sharing and community interaction. In those websites, users can ask their peers questions about their specific problems, or answer questions asked by others. Using those posts, community members can share information among themselves and provide a more approachable, personal and customized experience to those who need it. These websites, perhaps due to the faster and easier way with which they allow users to acquire information,

have become some of the most popular websites on the internet, receiving millions of accesses and posts each month.

Some Q&A websites cover a broad range of topics (e.g., Quora), while others cater to specific communities of users that share similar interests. Currently, there are Q&A websites covering a range of topics, from cooking[5] and photography[6] to academic research[7]. The large amounts of data generated by these websites are a valuable asset to understanding how users discuss these topics, share knowledge, and interact among themselves.

Software developers have become specially fond of Q&A websites, with many websites focusing on specific aspects of technology and software development. Stack Overflow, the most popular of these websites aimed at developers, currently holds millions of posts regarding varied topics about programming and technology, receiving 100 million monthly visitors and ranking among the 50 most popular websites in the world [82].

Many researchers have previously studied Stack Overflow and its many aspects [4]. For example, Barua et al. [12] have analyzed the topics discussed by developers, while Wu et al. [131] have explored how developers utilize the code discussed in it, and Bazelli et al. [13] have explored the personality traits of Stack Overflow users.

Researchers have also analyzed Q&A websites focused on discussing varied topics, such as health [49] and social [40] Q&A websites. Many of these studies searched for ways of improving these websites, such as helping users find information [90, 143] and facilitating their interactions [89, 108, 110, 126], and analyzed users' motivations for participating in the websites [21, 31, 35, 41, 55, 142].

Kamienski and Bezemer [57] were the first to analyze Q&A websites for game development. In their study, they found that three of the largest of those Q&A websites, namely Unity Answers, the UE4 AnswerHub, the Game Development Stack Exchange

[5]https://cooking.stackexchange.com/, accessed September 6th, 2021.
[6]https://photo.stackexchange.com/, accessed September 6th, 2021.
[7]https://www.researchgate.net/topics, accessed September 6th, 2021.

were in decline, with a decrease in user activity over the past few years. They also found similar results for questions about game development on Stack Overflow. Their findings stress the importance of studying and improving those Q&A websites to provide a better and more effective community for game developers.

3.3.2 Duplicate document detection on websites

Detecting duplicate documents is an important task for many types of websites. In forums, social networks, and Q&A communities alike, the presence of multiple posts with the same or closely related content may flood the website and hinder the ability of its users of finding the information they want. This effect is similar to spamming, and can also be detrimental by increase the amount of resources the websites have to dedicate to deal with the issue.

Many of the websites that suffer from having multiple duplicate posts have a system of manual duplicate detection. In those websites, users have to manually tag duplicate content, while referencing the original ones. Another common approach is to have moderators manually filter and approve posts before they can be published. These manual approaches require a lot of effort from users and moderators.

With that in mind, several researchers have looked into ways of relieving the burden of manually identifying duplicate documents by proposing automated duplicate document detection techniques [23, 67]. Specifically in the software engineering context, researchers have invested great effort into studying and developing automated detection techniques for duplicate pull requests [61, 62, 123] and bug reports [46, 47, 92, 93].

Software engineering researchers have also focused on studying duplicate questions on Stack Overflow. While some studies have analyzed different aspects of duplicate questions (e.g., their main characteristics [33] and impacts in the community [1]), the majority of work focused on developing systems to automatically detect duplicate questions. For example, Zhang et al. [144] introduced DupPredictor, which uses

title, description, topic and tag similarities to identify duplicates. Building on that, Ahasanuzzaman et al. [3] described Dupe, which increased the performance of the previous approach by using a two-step ranking system and a different set of question similarity metrics.

Others have since improved on those results by using different sets of techniques and similarity measures. Zhang et al. [138, 139] have achieved a higher performance when detecting duplicates by introducing several new features (e.g., features based on Doc2Vec and association rules). More recently, Wang et al. [121, 122] could also achieve a higher performance when using different neural network architectures, such as Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN).

Other studies have sought to aid and improve the development of duplicate question detection techniques by creating and analyzing features, methodologies, and the quality of labelled datasets [50, 74, 111, 132, 140, 141].

Researchers have also tackled the problem of detecting duplicate questions on other Q&A websites such as Quora or those belonging to the Stack Exchange network[8] [48, 52, 88, 101, 107, 146]. Studies have also explored how to use domain adaptation to create duplicate detection techniques with no labelled data [56, 63, 86, 100, 104, 109, 133]. As far as we know, no other studies have analyzed duplicate questions about game development, or made any efforts to develop and evaluate duplicate detection models for this specific domain.

3.4 Methodology

In this section we describe our methodology for collecting and processing the data used in our study, and applying techniques for duplicate question detection. Figure 3.1 shows an overview of the steps we have taken in this methodology. The code, data, and models used in this study are available online in our replication package[9] to allow

[8] https://stackexchange.com/, accessed September 6th, 2021.

[9] Our replication package is available online at https://github.com/asgaardlab/done-21-arthur-duplicate_gamedev_questions-code.

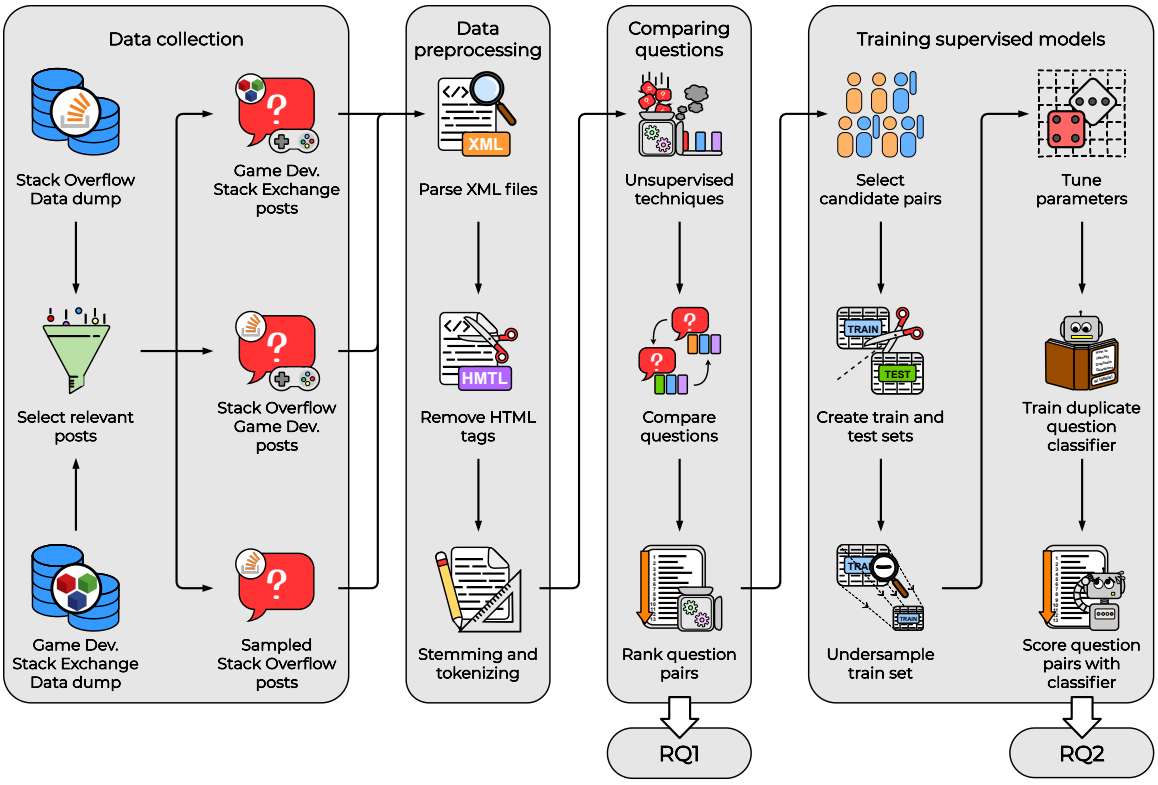


Figure 3.1: Overview of the steps we have taken in our methodology.

future researchers to use, reproduce, and evaluate our results.

3.4.1 Data collection

In this study, we used two sets of game development questions that were extracted from the Game Development Stack Exchange and Stack Overflow to test and evaluate the performance of the duplicate question detection techniques. We chose these two websites as they are the largest sources of labelled game development Q&A data. While part of our methodology is not reliant on labelled data, we still need these labels to evaluate our approach. Other larger websites such as Unity Answers and the UE4 AnswerHub do not offer a duplicate tagging feature, hindering our ability to use their data for our analyses. We also used a third set of general software development questions from Stack Overflow as a way of analyzing how the techniques perform on a different domain.

We collected the three sets of questions from the June 2021 Stack Exchange data

dump[10]. To build these question sets, we first downloaded all of the posts (i.e., questions, answers, and comments) for Stack Overflow and the Game Development Stack Exchange, along with the lists of relationships between posts.

From the two initial datasets, we selected the questions and answers from the list of posts. We also selected only the duplicate question relationships from the list of all post relationships, excluding those relationships that contain references to deleted questions that are not present in the datasets.

We followed the same methodology as Kamienski and Bezemer [57] to create the set of questions about game development from Stack Overflow. Thus, we selected the questions from the set of Stack Overflow posts by searching for questions marked with one of the following tags: *'game-engine'*, *'game-physics'*, *'game-development'*, *'gameobject'*, *'2d-games'*, *'unrealscript'*, *'unreal-engine4'*, *'unreal-blueprint'*, *'unreal-development-kit'*, *'unityscript'*, *'unity-ui'*, *'unity-editor'*, *'unity-networking'*, *'unity-webgl'*, *'unity5'*, *'unity5.3'*, *'unity3d'*, *'unity3d-5'*, *'unity3d-mecanim'*, *'unity3d-unet'*, *'unity3d-2dtools'*, *'unity3d-gui'*, *'unity3d-terrain'*, *'unity3d-editor'*, *'unity2d'*.

We created our third set of questions by sampling a small number of questions from Stack Overflow. Despite containing questions about multiple topics other than game development, this dataset allows us to test our approaches on a different set of data and analyze how they perform on a distinct yet related domain. We chose to sample a similar number of questions to those in the other datasets to eliminate the risk of performance variations caused by disparate amounts of data. Therefore, we randomly selected a number of questions equal the mean number of game development questions in the other two datasets mentioned above. We used the same approach to randomly sample duplicate question pairs, thus maintaining a similar proportion of duplicates across all datasets. We performed this process five times using different random seeds to obtain five distinct samples with the same number of questions and duplicate pairs. For the remainder of this chapter we treat the five samples as a single

[10]https://archive.org/details/stackexchange, accessed September 6th, 2021.

dataset and report the mean and standard deviation obtained from the samples for each of our results.

Table 3.1 shows the summary of each of the three datasets of questions we used in our study. We defined a duplicate question as a question that has a duplicate relation with another one in the dataset, in a unidirectional relationship. We refer to the questions referenced by duplicate questions as main questions. Duplicate questions point to one or more main questions[11], forming a duplicate question pair.

Table 3.1: Summary of the three datasets used in our methodology. Duplicate questions are defined as questions that have a duplicate relation with others. Each duplicate question forms one or more pairs with other questions of the dataset. The percentages are show in relation to the total number of questions in each dataset.

Website	Topic	Questions	Non-duplicates	Duplicates	Pairs
Stack Exchange	Game development	51,797	50,694	1,103 (2.1%)	1,144
Stack Overflow	Game development	68,200	67,191	1,009 (1.5%)	1,070
	General development	59,998	58,891	1,107 (1.8%)	1,107

3.4.2 Data preprocessing

The data we collected from the Stack Exchange data dump is in raw XML format and needs to be preprocessed before being used in our study. First, we parsed the XML files to extract the title, body, and tags for each of the questions in the dataset, which comprise all of the textual information provided by the question author at the time of posting. We also exclude from our dataset questions with no text as they may harm our future analyses.

We selected the accepted answer for each answered question in our dataset to be used when comparing questions in Section 3.4.3. If the question had no accepted answers, we chose the one with the largest number of votes that was posted first.

[11] Although duplicate questions usually point to only one main question, some duplicates point to several others. For example, question 10661714 links to five main questions: https://stackoverflow.com/questions/10661714. However, over 95% of duplicate questions have only one main question in our datasets.

Other studies have also used the number of votes received by answers as proxies for their quality [30, 78, 91]. We extracted the answer bodies from the selected answers and matched them to their corresponding questions.

We processed all the text elements we collected (i.e., question titles, bodies, tags, and answers) by removing any HTML tags and replacing any references to code snippets, images, and URLs with unique token identifiers. As the techniques used in this study are intended for natural language, these elements may degrade their results [12, 111, 116, 134]. Using regular expressions, we identified URLs contained in <a> tags or following a pattern of contiguous strings of characters preceded by *http://* or *https://* and replaced them with tokens. We also used regular expressions to replace any content between <code> and <img> HTML tags with tokens. We used Python 3's Beautiful Soup 4 library [99] to completely remove other HTML tags and elements.

As a final preprocessing step, we applied the `text_preprocess` function provided by Python 3's Gensim library [97] to remove punctuation, multiple whitespaces, numeric characters, stopwords, and short words. The function also stems the texts using the Porter Stemmer [87] and tokenizes them by splitting words separated by spaces.

3.4.3 Comparing questions

To identify if a question is a duplicate of another one, we need ways of comparing them and analyzing how similar they are. Researchers have proposed several methodologies to measure question similarity, ranging from a simple matching of co-occurring terms [3, 144], to more complex deep-learning-based techniques [121, 122]. In this study, we analyze how seven techniques perform on the task of identifying duplicate questions about game development. Two of those techniques have not yet been used for the task of detecting duplicate questions in a software engineering domain.

We chose techniques that range in complexity and computational cost, as a way of identifying those that are more cost-effective for our task. We only use unsupervised

and pre-trained techniques as they do not demand labelled data and can be implemented with relatively few computational resources. While custom techniques created to perform specific tasks can achieve higher performance on their domains, the cost of implementing them is high (both in terms of amount of data and computation), and they may not be a feasible alternative for some websites.

Table 3.2 presents a summary of the techniques we used, indicating the ones that have been previously used for detecting duplicate questions in software engineering domains. We used these techniques to produce similarity scores between the questions in our three datasets. Aside from BM25, which already produces a similarity score between two documents as its output, all of the other techniques convert the input documents to vectors of real numbers. We then compared these vectors using the cosine similarity (or the Jensen-Shannon divergence for probability distributions) to obtain a similarity measure between the documents. We used these similarity measures as a proxy for the likelihood of two questions being duplicates. We analyzed each of these similarity measures separately to identify the best one in the task of identifying duplicate questions.

Table 3.2: Summary of the techniques for duplicate question detection we used in our study. The *Pre-trained* column shows the techniques that require a training step prior to duplicate question detection. The *Supervised* column shows if a technique requires a labelled set of data during training.

Technique	Pre-trained	Supervised	Prev. used in Soft. Eng.	Used in
Jaccard	No	No	Yes	[111, 144]
TF-IDF	No	No	Yes	[138, 139]
BM25	No	No	Yes	[3, 111, 139]
Topic	Yes	No	Yes	[111, 138, 139, 144]
Doc2Vec	Yes	No	Yes	[138, 139]
BERTOverflow	Yes	No	No	
MPNet	Yes	Yes	No	

We also analyzed how using different text elements from questions affect the per-

formance of the techniques in Table 3.2. Figure 3.2 shows an overview of the methodology we used for comparing the questions using these different elements. We used a similar approach to other studies [3, 111, 139, 140, 144] to create five documents using question titles, bodies, and tags individually, the junction of titles and bodies, and the junction of titles, bodies, and tags. We also introduce a new comparison between all of the elements (i.e., title, bodies, and tags) with the addition of their answers. For duplicate questions that do not have answers, we only use their titles, bodies, and tags for this comparison. Other studies have shown that answers can be useful for detecting duplicate questions [1, 63], and we thus evaluate the impact of their usage in our methodology.

We applied each technique to each of these documents, obtaining 42 different similarity measures for each question pair (6 documents $\times$ 7 techniques). We performed this process for comparing each duplicate question to every other answered question in the dataset, as main questions need to have at least one answer to be referenced by a duplicate[12]. We provide further explanations of the comparison techniques we used in the following sections.

Jaccard similarity

The Jaccard similarity coefficient [53] is a common metric for measuring the similarity between two mathematical sets and is frequently used in information retrieval systems as a way of comparing two documents [79]. Before calculating the Jaccard similarity between two texts, we converted them into sets by selecting the unique tokens contained in each of them. Then, we calculated the metric using the equation $Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$, where A and B are the sets of tokens for two documents A and B.

[12]https://stackoverflow.com/help/duplicates, accessed September 6th, 2021.

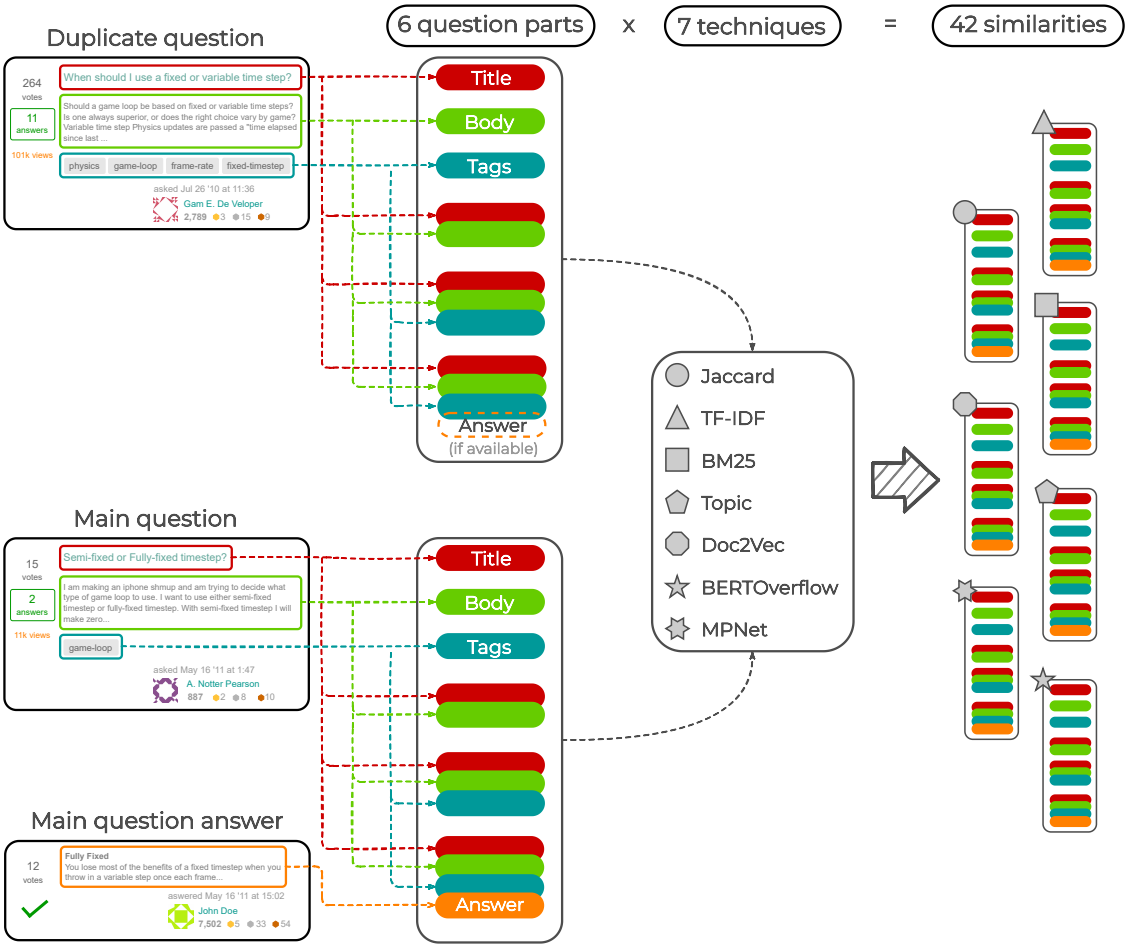


Figure 3.2: Overview of the methodology we used for comparing questions using seven techniques.

TF-IDF similarity

Term frequency-inverse document frequency (TF-IDF) is a technique for defining the relevance of a word in a document relative to all of the other documents in the corpus. Despite its simplicity, TF-IDF is used in many information retrieval applications and has been shown to be an effective technique for comparing documents [94]. In our study, we used the `TfidfVectorizer` implementation provided by Python's scikit-learn library [85] with default parameters for converting the documents of our corpora into vectors of TF-IDF values. In this implementation, the TF-IDF value of term t in a document d is given by the equation

$$TF\text{-}IDF(t,d) = TF(t,d) \times \log\left(\frac{n}{DF(t)+1}\right),$$

where $TF(t,d)$ is the number of times term t appears in document d, n is the number of documents in the corpus, and $DF(t)$ is the number of documents that contain term t. We computed the TF-IDF vectors separately for each of the six documents from each question in each dataset. We then computed the cosine similarity between two TF-IDF vectors to obtain a TF-IDF similarity measure using scikit-learn's `cosine_similarity` function.

BM25

BM25 is a ranking function that takes into account the frequency of each token in the corpus and in each document of the corpus to assign a score to a pair of documents. We used our own custom implementation of the BM25 algorithm derived from the one provided in Gensim 3.8[13]. Based on the study of BM25 parameters for duplicate question detection on Stack Overflow performed by Ahasanuzzaman et al. [3], we defined the values of the free parameters k_1 and b as 0.05 and 0.03, respectively.

[13]https://github.com/RaRe-Technologies/gensim/blob/3.8.3/gensim/summarization/bm25.py, accessed September 6th, 2021.

Topic similarity

We measured the similarity between the topics of two documents using the latent Dirichlet allocation (LDA) algorithm [15]. This unsupervised algorithm assumes that topics are represented by distributions of words to compute the probability of a document belonging to a topic. We trained LDA models based on our datasets using Gensim's implementation given by the `LDAModel` class. We trained one model for each set of documents extracted from questions from each dataset, totalling 18 different models (6 document sets $\times$ 3 datasets). We used these models to calculate vectors of topic probabilities for each question in our datasets. The main parameters that control the output of the algorithm are *alpha* and *eta*, which we set to `symmetric` and `auto`, respectively. We set the number of topics to 30, which is also the value used in other studies [139, 140]. Finally, we calculated the Jensen-Shannon divergence to measure the similarity between two vectors of topic probabilities using the `jensenshannon` function from Python's Scipy package [120]. We chose to use the Jensen–Shannon divergence for these comparisons as it is a more appropriate metric for calculating the similarity between two probability distributions when compared to the cosine similarity used in other studies.

Doc2Vec similarity

Doc2Vec [60] is an unsupervised algorithm based on Word2Vec [72, 73] for representing documents as fixed-length vectors of numbers. These vectors are created in a way such that two semantically similar documents are closer apart in the multi-dimensional space than two semantically different ones. We used Gensim's `Doc2Vec` class to train Doc2Vec models based on our data and compute vectors for the documents in our datasets. We trained one model for each set of documents in our three datasets, obtaining 18 models (6 document sets $\times$ 3 datasets). We used the same parameters as indicated by Zhang et al. [138], which are shown in Table 3.3. We compared the vectors obtained from the algorithm using scikit-learn's `cosine_-`

`similarity` function to produce a similarity measure between the documents.

Table 3.3: Parameters used for learning document embeddings using Gensim's Doc2Vec implementation, following suggestions from Zhang et al. [138].

Parameter	Value	Description
vector_size	100	Number of dimensions of the feature vector
window	15	Maximum distance between the current and predicted words
min_count	1	Minimum frequency required for words before being ignored
sample	1e-5	Threshold for downsampling high-frequency words
negative	1	Number of "noise words" drawn when negative sampling
epochs	100	Number of passes over the training corpus
seed	42	Seed number for the random number generator

BERTOverflow similarity

BERTOverflow [116] is a model for producing word embeddings that is pre-trained on 152 million sentences collected from Stack Overflow's data dump. This model is based on BERT [32], a deep neural network-based algorithm for producing vector representations of words which can be expanded and fine-tuned for different natural language processing tasks. Unlike Word2Vec, BERT uses the context in which words are used to create more accurate vector representations. Despite not being trained specifically for the game development domain or for detecting duplicate questions[14], we used BERTOverflow in the hope that some of the knowledge it acquired from training on Stack Overflow can be used for our intended application. We used the pre-trained model provided by the authors in Python's Transformers package [130][15] with default parameters, and adapted it to produce document vectors using the SentenceTransformers [98][16] package. We performed no other training steps to tune

[14]BERTOverflow was originally created for code and named entity recognition, but its word embeddings can be used for many natural language processing tasks.

[15]https://huggingface.co, accessed September 6th, 2021.

[16]https://www.sbert.net, accessed September 6th, 2021.

the model. As the model implements its own tokenization function, we used the untokenized documents extracted from questions to produce sentence embeddings. We compared the document vectors produced for each document using scikit-learn's `cosine_similarity` function to produce a similarity measure between the documents.

MPNet similarity

MPNet [113] is another deep neural network-based model for creating word embeddings for natural language processing tasks. It uses permuted language modelling and token position information to obtain an increased performance when compared to other BERT-based models. In our study, we used the `paraphrase-mpnet-base-v2` model provided by the SentenceTransformers [98] package, which is based on MPNet, to produce document vectors. This model was pre-trained and fine-tuned for the task of producing document vectors, and currently offers the best average performance on a set of document comparison tasks[17]. We used the model with default parameters and did not perform other training or tuning steps. Similar to what we did with BERTOverflow, we used untokenized documents for creating sentence embeddings based on the documents extracted from questions. Once again, we compared the document vectors using the `cosine_similarity` function provided by scikit-learn to produce a similarity measure between the documents.

3.4.4 Training supervised classifier models

The question comparison techniques we described in Section 3.4.3 provide similarity scores that help us in identifying duplicate questions. However, each technique takes a different approach to determining how similar two documents are and provide different views on the task of detecting duplicates. Therefore, we sought to merge the similarity measures described above into a single score that can hopefully combine these views and achieve higher performance on the task of detecting duplicate game development

[17]https://www.sbert.net/docs/pretrained_models.html, accessed September 6th, 2021.

questions.

Figure 3.1 shows an overview of the methodology we used to obtain the new similarity measure. This new measure is the output of a supervised classifier model that uses the 42 other measures as features and tries to predict whether a pair of questions are duplicates, in an approach similar to that used by Zhang et al. [139]. We calculate the new measure by first applying the techniques described in Section 3.4.3 to extract the similarity scores from a pair of questions and then using the classifier to make a final prediction. We leveraged the small set of labelled duplicate pairs provided by our datasets to train these classifiers.

Instead of using all possible question pairs to train and evaluate the models, we limited the number of pairs by selecting a number C of candidate questions for comparison with each duplicate question in the datasets. The selection of candidate questions was proposed by Ahasanuzzaman et al. [3] as a way of reducing the computational cost of identifying duplicate questions. The authors of that study selected the 10,000 most relevant candidate questions for each duplicate using BM25 prior to computing the final similarity score using their custom technique. As far as we know, the authors did not use any other techniques for improving search speed such as inverse indices. In our study, we compared the different similarity measures we obtained in Section 3.4.3 and experimented with several different values for C to identify the set of parameters that produces the best result.

We note that selecting candidates prior to training and evaluating the model can introduce bias towards the metric used for choosing relevant candidates. However, this process also reduces the proportion of duplicate to non-duplicate question pairs, which may aid the supervised model during training and scoring.

Prior to training the model, we randomly split the set of all candidate question pairs into train and test sets. We performed the split by defining 20% of the duplicate questions as test duplicates and assigning all of the candidate pairs composed by them to the test set, while assigning the remaining pairs to the train set. We made sure

to exclude any reference to test duplicate questions from the train set (i.e., candidate question pairs in which the duplicate question plays the role of a candidate), to avoid leaking test information into the train set.

To provide the model with additional examples of unrelated questions, we included 20% of fake candidate pairs in the train set. We selected those fake candidate pairs by applying the same process used for duplicate questions to a set of randomly selected questions. As the number of duplicate question pairs is tiny when compared to the number of non-duplicate pairs, we undersampled the majority class of non-duplicate pairs in the train set by randomly selecting non-duplicate pairs while maintaining the duplicate ones, reaching a proportion of 1 true duplicate pair to 99 non-duplicate pairs. We experimented using a larger number of duplicate pairs to non-duplicate pairs, but found that they reduced the performance of the classifier models. We did not undersample or alter the number of candidate pairs in the test set to simulate real world conditions.

Finally, we trained the classifier models using Random Forests provided by scikit-learn's `RandomForestClassifier` class. We trained the model using the similarity measures generated by the techniques in Section 3.4.3 for the question pairs in the train set, trying to predict whether the pairs are duplicates or not. The Random Forest models output a probability of the two questions in a pair being duplicate, which we use as a measure of the similarity between the two questions, in a similar fashion to the other techniques we discussed above.

We decided not to experiment with other types of supervised models as the results reported by other researchers did not show great differences in performance when changing the models used for this task [121, 122, 138, 139]. Instead, we chose to use the random forest algorithm as it obtained some of the best results in other studies [138, 139] that used it, while still being relatively simple.

We tuned the hyperparameters of the random forest to improve its performance in each individual train set using scikit-learn's `RandomizedSearchCV`. We ran 30 itera-

tions of a random parameter search with 5-fold cross validation. We used the same approach as we did to separate the train and test sets to create custom folds for the search, thus making sure that no information is leaked across folds. We trained the final Random Forest model using the whole train set and the best parameters found during search. Finally, we scored the test set of candidate pairs using the trained models, obtaining a measure for their likelihood of being duplicates of one another.

3.5 Results

In this section we discuss the motivation, approach and findings for each of our research questions. Section 3.5.1 discusses the performance of the similarity scores obtained from the techniques described in Section 3.4.3, while Section 3.5.2 analyzes the performance of the supervised classifier models described in Section 3.4.4.

3.5.1 RQ1. What is the performance of unsupervised and pre-trained techniques for duplicate question detection on game development Q&A data?

Motivation: There are several techniques for identifying duplicate questions. While some of those techniques require labelled data to learn to identify duplicate documents for specific applications, others can be used with no other information aside from the text contained in those documents. These techniques are usually unsupervised or pre-trained on different sets of data, and are specially useful when little to no labelled data is available, such as in the case of Unity Answers and the UE4 AnswerHub, the two largest Q&A websites for game development. Another advantage to those techniques is that they do not need large corpora or vast computational resources to function, and can thus be applied in situations where those two factors are a constraint. In this research question, we analyze the performance of seven of those unsupervised and pre-trained techniques on the task of identifying duplicate questions on two datasets of game development questions collected from Stack Overflow and

the Game Development Stack Exchange. By understanding how well those techniques can detect duplicate questions, we can determine how suitable they are for usage in Q&A websites as alternatives to more complex techniques that may not be easily implemented.

Approach: We used the similarity scores described in Section 3.4 to create different ranks for all of the question pairs produced from comparing duplicate questions to other answered questions in the dataset. Figure 3.3 shows an overview of our approach for evaluating these similarity scores. For a given duplicate question, we ordered the question pairs using each of the 42 similarity scores we obtained from comparing 6 question pairs using 7 techniques. We assigned an increasing rank number to each of the question pairs, with tied pairs being assigned the average rank of the tied group, obtaining 42 different ranks (one for each similarity). We used these ranks as a proxy for how well the similarity scores can identify true duplicate pairs. Better performing scores should assign top ranks to true duplicate pairs, while keeping false duplicate pairs at the bottom of the ranking.

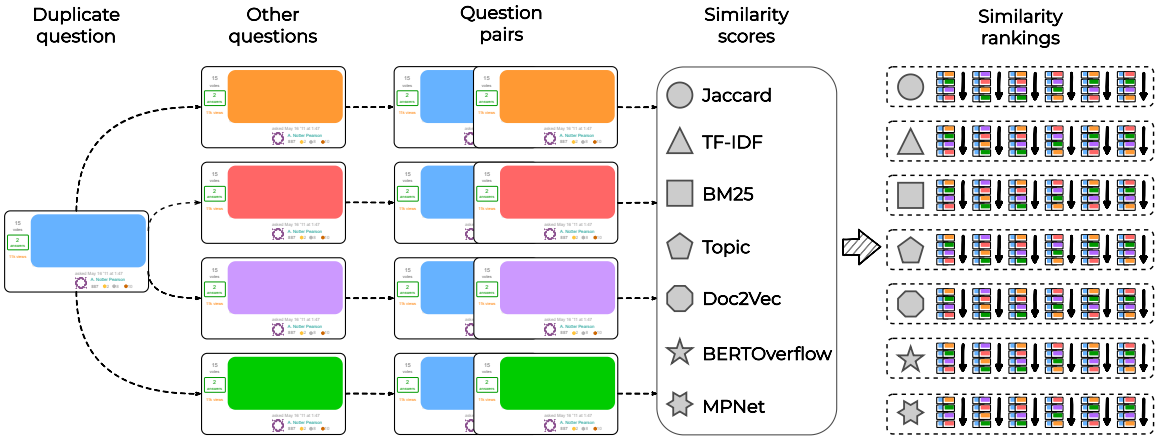


Figure 3.3: Overview of our approach to ranking question pairs according to similarity scores.

We used the recall-rate@k measure to analyze how each similarity score performs at providing a limited number of suggestions for possible duplicate questions. This measure was used in a number of other studies [3, 105, 111, 114, 115, 121, 122, 139,

144], and evaluates the ability of a score of accurately recommending a true duplicate question among a list of the top-k highest scores, simulating a real-world search system. In our study, we defined the recall-rate@k as the percentage of duplicate questions that have at least one true duplicate question pair ranked among the top-k recommendations. We used the following equation to calculate the recall-rate@k measure for each of the similarity score rankings mentioned above:

$$recall\text{-}rate@k = \frac{\sum_{i=1}^{N} v_i}{N},$$

where N is the number of duplicate questions in the dataset and v_i is a binary variable indicating if a duplicate question has at least one true duplicate pair with a rank of k or lower. We used values of k equal to 5, 10 and 20 in our analysis, as these are the most commonly used values in other studies.

Findings: **The techniques performed worse when detecting game development questions than when detecting general development questions on Stack Overflow.** Table 3.4, Table 3.5, and Table 3.6 show the performance of the 42 similarity scores we tested for identifying duplicate questions measured with different metrics. All of the similarity scores had lower performance at detecting duplicate questions from the two game development datasets than questions about general development from Stack Overflow. While the drop in performance was not as pronounced for the Game Development Stack Exchange, there was a large difference when detecting duplicate questions about game development from Stack Overflow. Nevertheless, up to 29% of the duplicate questions about game development were correctly ranked by the techniques.

Similarities calculated with MPNet achieved the highest performance. The similarities calculated using MPNet were the most effective when ranking question pairs from the Game Development Stack Exchange and about general development on Stack Overflow. On those datasets, the technique could correctly classify 40% and 50% of duplicate question pairs among the top 5 most similar pairs, respec-

Table 3.4: Performance of the studied techniques according to different metrics on the Game Development Stack Exchange. Columns 1 to 5 indicate the question part that was used for comparing the questions, as such: 1 - *title*, 2 - *body*, 3 - *tags*, 4 - *title* and body, 5 - *title*, *body*, and *tags*, and 6 - *title*, *body*, *tags*, and *answers*. Values in bold show the best scores we obtained for each metric in each dataset. Values in parentheses show the standard deviations we obtained for the five samples of the dataset about general development on Stack Overflow.

Metric	Technique	1	2	3	4	5	6
recall-rate@5	Jaccard	15.41	11.51	4.71	14.96	16.23	15.41
	TF-IDF	18.04	17.14	6.07	25.39	27.11	27.02
	BM25	16.95	12.33	7.07	15.50	16.95	20.85
	Topic	0.00	1.09	0.91	1.27	1.00	1.45
	Doc2Vec	0.00	1.27	0.00	1.27	1.72	1.00
	BERTOv.	5.26	2.99	4.71	4.90	5.62	6.26
	MPNet	27.20	25.20	7.07	37.72	37.81	**39.62**
recall-rate@10	Jaccard	20.13	14.05	7.80	18.13	19.58	20.76
	TF-IDF	21.94	22.94	9.25	31.01	33.64	33.54
	BM25	22.21	15.23	9.61	20.40	21.94	26.56
	Topic	0.54	1.54	1.00	1.45	2.18	2.27
	Doc2Vec	0.00	1.72	0.00	1.36	2.90	1.99
	BERTOv.	6.53	3.45	6.07	5.62	6.17	7.62
	MPNet	33.45	30.92	9.97	44.79	**46.33**	45.87
recall-rate@20	Jaccard	25.11	17.32	11.51	22.39	25.29	24.57
	TF-IDF	27.38	27.83	13.78	38.17	41.61	41.98
	BM25	27.74	20.04	14.51	26.56	28.74	34.72
	Topic	0.73	1.99	1.27	1.99	3.26	3.35
	Doc2Vec	0.00	2.18	0.00	2.72	4.35	3.08
	BERTOv.	7.62	3.81	8.16	6.71	7.71	9.61
	MPNet	39.80	37.26	14.14	52.86	53.31	**53.94**

Table 3.5: Performance of the studied techniques according to different metrics on the dataset about game development on Stack Overflow. Columns 1 to 5 indicate the question part that was used for comparing the questions, as such: 1 - *title*, 2 - *body*, 3 - *tags*, 4 - *title* and body, 5 - *title*, *body*, and *tags*, and 6 - *title*, *body*, *tags*, and *answers*. Values in bold show the best scores we obtained for each metric in each dataset. Values in parentheses show the standard deviations we obtained for the five samples of the dataset about general development on Stack Overflow.

Metric	Technique	1	2	3	4	5	6
	Jaccard	8.92	6.64	1.68	8.62	8.42	7.83
	TF-IDF	11.00	7.63	2.38	10.80	11.20	15.76
	BM25	11.40	6.14	2.18	7.73	8.62	**16.15**
recall-rate@5	Topic	0.50	0.79	0.20	0.40	0.50	1.19
	Doc2Vec	0.00	0.50	0.00	0.79	0.59	0.89
	BERTOv.	3.67	1.88	1.68	2.97	3.27	3.17
	MPNet	13.58	8.72	2.78	14.87	14.77	14.97
	Jaccard	12.49	7.63	2.38	11.10	11.30	10.21
	TF-IDF	15.86	10.01	3.27	15.16	14.77	21.51
	BM25	14.27	7.53	2.87	10.21	11.30	**21.70**
recall-rate@10	Topic	0.59	0.99	0.30	0.59	0.69	1.78
	Doc2Vec	0.00	0.69	0.00	1.39	1.59	0.89
	BERTOv.	4.56	1.98	2.28	3.96	3.77	3.37
	MPNet	17.74	11.10	3.47	21.01	19.72	21.01
	Jaccard	16.65	10.21	4.26	14.07	14.17	11.99
	TF-IDF	21.21	13.78	4.36	18.93	19.72	28.84
	BM25	19.23	9.12	4.86	12.98	14.27	**29.34**
recall-rate@20	Topic	0.99	1.59	0.50	1.19	0.89	2.28
	Doc2Vec	0.00	0.79	0.00	1.59	2.28	1.49
	BERTOv.	5.65	2.28	3.47	4.56	4.36	4.06
	MPNet	23.09	15.46	5.45	26.76	26.26	26.66

Table 3.6: Performance of the studied techniques according to different metrics on the dataset about general development on Stack Overflow. Columns 1 to 5 indicate the question part that was used for comparing the questions, as such: 1 - *title*, 2 - *body*, 3 - *tags*, 4 - *title* and body, 5 - *title*, *body*, and *tags*, and 6 - *title*, *body*, *tags*, and *answers*. Values in bold show the best scores we obtained for each metric in each dataset. Values in parentheses show the standard deviations we obtained for the five samples of the dataset about general development on Stack Overflow.

Metric	Technique	1	2	3	4	5	6
recall-rate@5	Jaccard	24.82 (0.86)	10.30 (0.42)	13.08 (1.11)	17.40 (1.06)	22.20 (1.86)	20.96 (1.39)
	TF-IDF	29.00 (0.97)	17.43 (0.82)	13.76 (1.18)	30.62 (0.58)	37.00 (0.79)	40.70 (1.03)
	BM25	27.68 (1.38)	11.83 (0.36)	15.39 (1.54)	22.58 (0.78)	28.83 (1.51)	35.66 (1.33)
	Topic	0.41 (0.21)	0.40 (0.23)	1.15 (0.30)	0.61 (0.35)	0.68 (0.14)	0.77 (0.22)
	Doc2Vec	0.02 (0.04)	0.58 (0.23)	0.02 (0.04)	0.88 (0.27)	1.56 (0.25)	1.90 (0.28)
	BERTOv.	7.14 (1.35)	1.38 (0.37)	8.06 (0.35)	3.00 (0.49)	5.04 (0.68)	4.72 (0.58)
	MPNet	40.09 (0.57)	23.11 (1.54)	16.75 (0.69)	45.98 (1.04)	48.51 (1.09)	**50.14** (1.28)
recall-rate@10	Jaccard	30.28 (1.28)	13.01 (0.57)	17.24 (0.68)	21.64 (0.86)	27.70 (1.87)	25.17 (1.79)
	TF-IDF	34.42 (1.08)	21.59 (0.34)	18.37 (1.01)	36.49 (0.63)	44.03 (0.91)	48.35 (0.47)
	BM25	32.76 (1.56)	15.52 (0.39)	20.11 (1.51)	28.08 (0.92)	35.34 (1.43)	43.63 (1.67)
	Topic	0.74 (0.29)	0.67 (0.36)	1.52 (0.36)	0.94 (0.48)	1.37 (0.28)	1.45 (0.23)
	Doc2Vec	0.02 (0.04)	0.90 (0.21)	0.02 (0.04)	1.26 (0.43)	2.40 (0.46)	2.71 (0.38)
	BERTOv.	8.42 (1.26)	1.66 (0.37)	10.05 (0.48)	3.90 (0.47)	6.20 (0.65)	5.85 (0.68)
	MPNet	46.34 (0.99)	28.06 (1.77)	22.02 (1.09)	52.27 (0.56)	55.68 (0.54)	**57.32** (0.63)
recall-rate@20	Jaccard	35.25 (2.03)	16.01 (0.96)	22.71 (0.44)	26.50 (1.20)	33.21 (1.72)	30.59 (1.39)
	TF-IDF	39.75 (1.21)	26.63 (0.37)	24.16 (0.56)	42.85 (1.00)	51.18 (0.59)	56.12 (1.08)
	BM25	38.57 (1.74)	19.89 (0.44)	25.98 (0.85)	34.04 (0.99)	41.86 (1.17)	52.11 (1.28)
	Topic	1.08 (0.30)	0.86 (0.39)	2.44 (0.56)	1.68 (0.61)	2.48 (0.48)	2.40 (0.32)
	Doc2Vec	0.02 (0.04)	1.34 (0.36)	0.04 (0.05)	2.03 (0.61)	3.47 (0.56)	3.94 (0.33)
	BERTOv.	9.67 (1.44)	1.95 (0.36)	12.47 (0.50)	4.95 (0.74)	7.32 (0.48)	7.36 (0.78)
	MPNet	52.21 (1.28)	33.60 (1.68)	27.79 (0.50)	59.06 (0.29)	61.97 (0.94)	**63.61** (0.34)

tively. The technique had a lower performance on game development questions on Stack Overflow, having only classified at most 27% of duplicate questions among the 20 top ranked pairs.

Other noteworthy techniques are the Jaccard, TF-IDF, and BM25 similarities, which could correctly rank up to 40% of duplicate pairs among the top 5 most similar pairs. These techniques achieved the highest performance when ranking duplicate questions about game development from Stack Overflow, even surpassing the scores obtained by MPNet similarities. Despite achieving up to 15% lower scores than MPNet on the other two datasets, these are relatively simple techniques when compared to the other ones we tested, and may be useful if time and computational resources are a limiting factor when detecting duplicate questions.

We achieved the highest performance when comparing all question parts including answers. We found that merging the title, body, and tags of the questions into a single document before comparing questions yielded some of the best results for almost all of the metrics and similarities we tested. Moreover, we found that using only the title of the questions was also a good strategy for finding duplicates, even beating the performance of comparing all of the questions parts for some techniques. These results are consistent with other studies that compared different question parts to detect duplicate questions on Stack Overflow [3, 139, 144].

We also found that using the answers from other questions can greatly improve the performance of the techniques. In all datasets, the similarities calculated using answers achieved the best overall results. We observed an increase of up to two percent points on the dataset from the Game Development Stack Exchange and general questions on Stack Overflow when using answers for comparing question pairs. The increase in performance was even higher for game development questions on Stack Overflow, with a boost of up to 15 percent points for the similarities calculated using TF-IDF and BM25.

The fact that some techniques gained a large boost when we used answers for com-

paring question pairs may indicate that some questions are marked as duplicates as a way of referencing the answers posted in the other question, while the questions are not duplicates themselves. For example, question `86517`[18] on the Game Development Stack Exchange asks how to shoot bullets from a spaceship using C++ and is marked as a duplicate of question `86326`[19], which asks how to spawn enemies using Java. The user that identified the duplicates posted a comment noting that the solutions to both questions are similar, but changes should be made to accommodate the differences between the projects.

The Topic, Doc2Vec and BERTOverflow similarities achieved a low performance. On all of the datasets these similarities ranked fewer than 10% of the duplicate question pairs among the 20 most similar pairs. Other studies have also shown that topic similarities offer poor performance on the task of identifying duplicate questions on Stack Overflow [139, 140, 144]. However, our results differ from the ones found by Zhang et al. [139, 140], where Doc2Vec similarities obtained better performance. These differences may be in part due to the different sets of data used in our studies. We further discuss some of those differences in Section 3.6.

The fact that the similarities calculated with BERTOverflow showed poor performance can be a consequence of the absence of a fine-tuning step prior to computing sentence embeddings. BERTOverflow is a model for generating word embeddings trained on Stack Overflow data and we performed no other training steps to adapt it to the task of generating and comparing sentence embeddings, as was done for the MPNet model.

Similarity scores based on Jaccard, TF-IDF, BM25 and MPNet could correctly rank most duplicate question pairs among the 5% most similar pairs. Figure 3.4 shows the distribution of ranks assigned to true duplicate question pairs for all of the 42 similarity scores we tested on our three datasets. When using

[18]https://gamedev.stackexchange.com/questions/86517, accessed September 6th, 2021.
[19]https://gamedev.stackexchange.com/questions/86326, accessed September 6th, 2021.

Jaccard, TF-IDF, BM25 and MPNet similarities, we observed that at least 75% of the duplicate question pairs were ranked among the 2,500 most similar pairs. Given that we compared each duplicate question with over 50,000 questions for each dataset, that represents less than 5% of all of the question pairs we analyzed. Other techniques produced worse rankings, but could still place most duplicate pairs among the 20% (10,000) most similar pairs.

However, as we can see from Table 3.4, Table 3.5, and Table 3.6, these rankings do not necessarily translate to good performance on ranking duplicate pairs among the 20 most similar ones. Instead, these results show that these techniques can be used by themselves as heuristics for reducing the number of questions that have to be compared to identify their duplicates. Given that techniques such as the Jaccard, TF-IDF, and BM25 similarities have fast computation times, they may prove useful for a pre-selection step such as the one discussed in Section 3.4.4. A limited number of question pairs can help improve the performance of duplicate question detection systems by reducing the number of false duplicate pairs that are evaluated and reducing the time it takes to evaluate a set of questions.

Summary: The studied techniques could rank up to 54% of the duplicate question pairs among the 20 most similar pairs in datasets about game development. However, these techniques showed worse performance when ranking duplicate questions about game development than when ranking questions about general development on Stack Overflow. We achieved the best results by using MPNet similarities for comparing question titles, bodies, tags, and answers. Other techniques also had noteworthy performance and can be used for selecting candidate question pairs for improving the performance of duplicate question detection systems.

3.5.2 RQ2. How can we leverage labelled data to improve the performance of unsupervised techniques?

Motivation: In RQ1 (Section 3.5.1) we explored unsupervised and pre-trained techniques for the task of detecting duplicate game development questions. While these techniques do not require labelled data, our datasets contain a small set of labelled

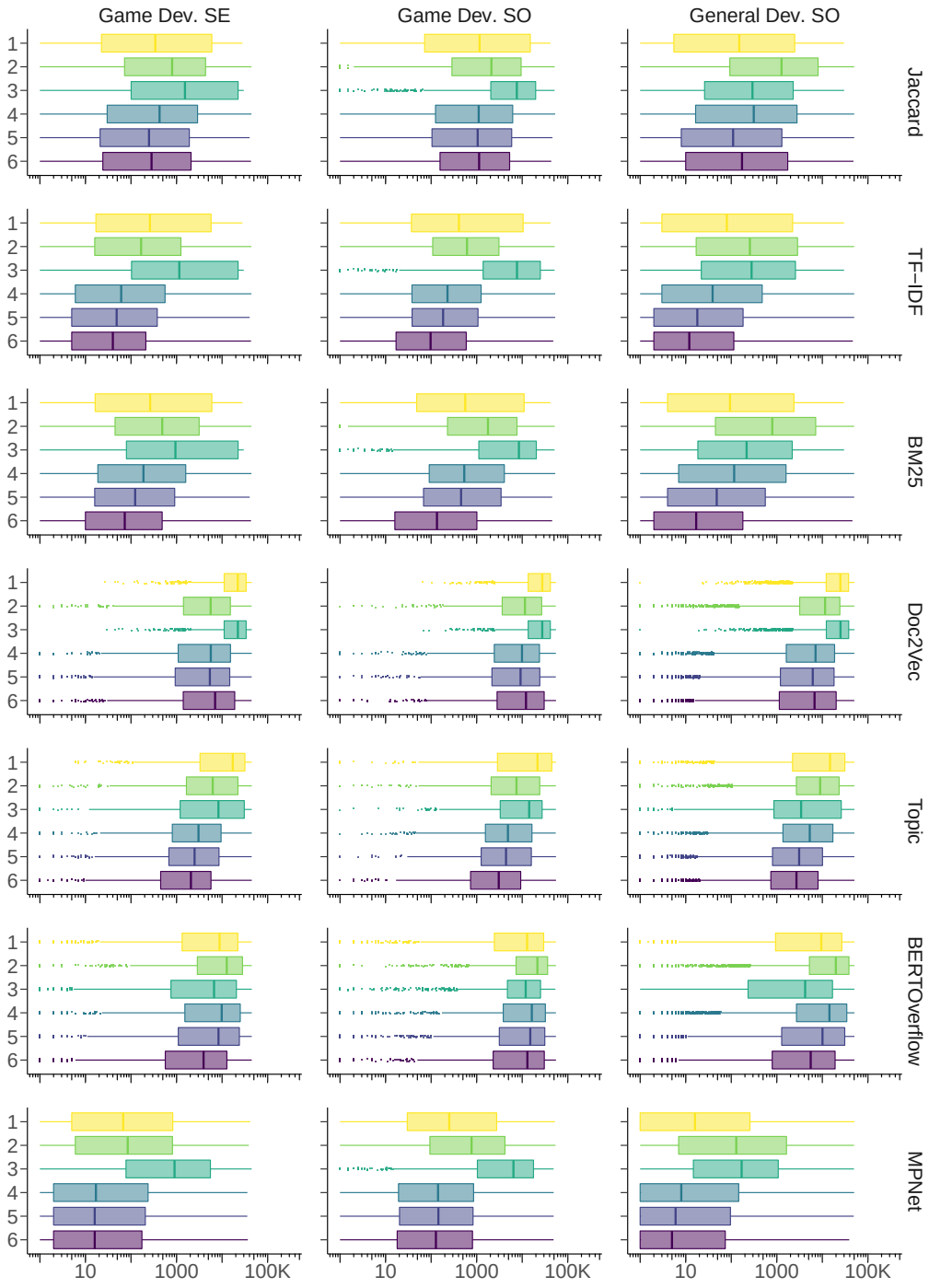


Figure 3.4: Distribution of ranks of true duplicate pairs in the three different datasets used in this study according to different techniques. Labels 1 to 5 indicate the question part that was used for comparing the questions, as such: 1 - *title*, 2 - *body*, 3 - *tags*, 4 - *title* and body, 5 - *title*, *body*, and *tags*, and 6 - *title*, *body*, *tags*, and *answers*.

duplicate question pairs that can be useful for learning the characteristics of duplicate questions. Furthermore, we have explored a set of 42 similarity scores that take different approaches to comparing questions, and aggregating them into a single score can be helpful to achieve a higher performance on the task of detecting duplicate questions. In this research question, we explore how we can combine these different scores and use the labelled duplicate question data to build a better similarity score that can more reliably detect duplicate questions.

Approach: We followed the methodology described in Section 3.4.4 to build a supervised classification model for detecting duplicate question pairs. We started by selecting candidate question pairs for each duplicate question in the datasets by selecting the top ranked pairs according to the TF-IDF similarity score between question titles, bodies and tags. We analyzed how the number of selected candidate pairs affects performance by training models with sets of various numbers of candidates.

As we discussed in RQ1 (Section 3.5.1), TF-IDF is a simple and fast technique that provided one of the best rankings for duplicate question pairs. Even though the MPNet similarity scores offered better performance, computing those similarities takes considerably more time as the vectors used for representing each document are much larger than the ones used for TF-IDF[20]. For example, it took around 10 seconds to compute TF-IDF vectors for each dataset using one core of an 8th generation Intel Core i7 processor, and another 180 seconds to compute their similarities. Meanwhile, it took around 360 seconds to compute MPNet vectors for each dataset using a Tesla P100 GPU, and another 270 seconds to compute their similarities on the laptop mentioned above.

We followed the methodology described in Section 3.4.4 to create train and test sets. We trained Random Forest models using the hyperparameters that provided the

[20]While the vectors produced by TF-IDF have as many dimensions as the size of the vocabulary in the corpus, these vectors are sparse and can only have as many non-zero values as the number of unique words in a document. In our datasets, the median number of unique words in each question is around 30. Meanwhile, MPNet produces dense vectors with a fixed size of 768 dimensions.

best recall-rate@5 for each set of candidates during the hyperparameter tuning stage. We then used the trained models to obtain similarity scores for the candidate pairs in the test sets. Similar to what we did in RQ1, we analyzed the final performance of the models using the recall-rate@k metric for values of k of 5, 10, and 20.

Given that we created models to identify duplicate question pairs on specific datasets, they might have learned particular characteristics of the questions on which they were trained. We thus analyze the performance of a model when identifying duplicate question pairs from other datasets than the one used for training it. For example, we use the model trained on the Game Development Stack Exchange data to identify duplicate questions on Stack Overflow, and vice-versa. As some of Q&A websites do not have the labelled data required to train supervised classifiers, a model trained on another set of data may prove a better alternative than using other techniques such as the ones we discussed in RQ1.

We investigated the duplicate question pairs that our classifier could not correctly rank among the top 20 results. First, we selected the set of misclassified pairs in the test set and checked if the true duplicate pair was among the set of candidate questions or if it was removed during the candidate selection process. Furthermore, we read both questions of the misclassified pairs to define a possible cause for the low ranking. We also read the top ranked question associated to each of those question pairs, to see if it was a possibly unlabelled duplicate pair. We judged a top ranked question as a correct pair for the duplicate question if it provided enough information to solve the problem being discussed. We performed this analysis for the datasets about game development from Stack Overflow and the Game Development Stack Exchange, and for one of the samples of the dataset about general software development on Stack Overflow.

Findings: **The optimal number of candidates depends on the dataset and metric being used.** Table 3.7 shows the performance of the classifier models we trained using different numbers of candidate pairs. We observed that the performance

of the classifiers varied according to the number of candidate question pairs used for training and evaluating them. Overall, using a larger number of candidates increased the performance of the classifiers in the recall-rate@10 and recall-rate@20 metrics, but decreased the recall-rate@5 metric. We found that the classifiers achieved the highest performance in the recall-rate@5 metric when using a number of candidates between 500 and 1,500. Meanwhile, a number of candidates between 2,000 and 7,500 achieved higher recall-rates at 10 and 20.

Therefore, we recommend using a number of candidates in the range of 1,000 to 7,500. We note that the performance of the classifiers trained using that range showed only small differences in performance, and that using larger numbers of candidates can increase the time it takes to evaluate each duplicate question. We used a number of candidates of 1,500 for our other analyses, as it showed a good performance in all datasets and recall-rates.

We could almost double the performance of the unsupervised techniques by using supervised classifiers. Table 3.7 highlights the best scores achieved by our models in bold. We observed a performance increase of up to 14 percent points in the recall-rate@5 for the model trained on game development questions from Stack Overflow (from 16.15 to 30.20), representing an 88% increase when compared to the best similarity scores we obtained in RQ1 (Section 3.5.1). We also observed a performance increase of 20% for the models trained on general questions from Stack Overflow, which could correctly classify 60% (against 50% in RQ1) of duplicate questions among the 5 most similar question pairs. We observed similar increases in the other recall-rate metrics for these two datasets. The classifiers trained on the Game Development Stack Exchange showed more modest increases of up to 9%.

The models could predict duplicate questions on other datasets with little to no decrease in performance. Table 3.8 shows the performance of models trained with 1,500 candidates when detecting duplicate questions on datasets other than those used for training them. We found that the models trained on game develop-

Table 3.7: Performance of the duplicate question classifier models for different numbers of candidate pairs according to different metrics. Values in bold show the best results obtained for a metric in a given dataset. Values in parentheses show the standard deviations we obtained for the five samples of the dataset about general development on Stack Overflow.

Dataset	Candidates	Recall-rate@		
		5	10	20
Game Dev. Stack Exchange	100	39.37	46.15	51.58
	250	38.01	47.51	54.75
	500	36.20	44.80	55.66
	750	38.01	49.77	**58.82**
	1000	38.46	46.15	56.11
	1500	40.27	**50.68**	**58.82**
	2000	39.82	47.51	56.11
	2500	**41.18**	49.32	57.01
	5000	40.72	50.23	**58.82**
	7500	38.91	49.32	56.56
	10000	38.46	47.96	**58.82**
Stack Overflow/Game dev.	100	**30.20**	37.13	46.53
	250	29.21	38.12	47.52
	500	25.25	35.15	43.07
	750	29.70	39.11	48.51
	1000	28.22	**40.10**	48.51
	1500	27.23	38.12	**50.00**
	2000	28.22	38.12	48.51
	2500	28.22	38.61	48.02
	5000	29.70	38.61	47.03
	7500	28.22	38.61	49.01
	10000	29.70	38.12	47.52
Stack Overflow/General dev.	100	58.28 (3.60)	64.34 (2.16)	68.42 (1.79)
	250	59.64 (3.58)	64.70 (3.33)	70.59 (2.19)
	500	59.91 (5.16)	65.70 (4.48)	72.13 (3.28)
	750	59.82 (4.48)	66.16 (3.86)	71.40 (3.67)
	1000	**60.00** (3.75)	65.79 (4.13)	72.13 (2.61)
	1500	59.55 (5.12)	65.88 (3.98)	**72.49** (3.08)
	2000	58.82 (4.34)	65.88 (4.75)	72.31 (2.99)
	2500	59.00 (4.75)	65.61 (3.75)	72.22 (3.19)
	5000	59.10 (4.64)	**66.43** (3.78)	72.13 (3.30)
	7500	55.93 (7.98)	64.16 (5.16)	71.68 (3.13)
	10000	57.83 (4.47)	65.16 (3.72)	72.13 (2.53)

Table 3.8: Performance of the duplicate detection models in cross-dataset settings.

Test dataset	Train dataset	Recall-rate@ 5	10	20
Game Dev. Stack Exchange	Game Dev. Stack Exchange	40.27	50.68	58.82
	Stack Overflow/Game dev.	35.75	44.80	53.39
	Stack Overflow/General dev.	39.00 (2.03)	48.60 (1.30)	55.84 (0.41)
Stack Overflow/Game dev.	Game Dev. Stack Exchange	26.73	35.15	47.03
	Stack Overflow/Game dev.	27.23	38.12	50.00
	Stack Overflow/General dev.	25.54 (0.57)	33.27 (0.81)	45.05 (1.68)
Stack Overflow/General dev.	Game Dev. Stack Exchange	58.01 (3.42)	63.98 (2.78)	69.77 (3.78)
	Stack Overflow/Game dev.	52.76 (4.79)	61.00 (5.38)	68.14 (4.43)
	Stack Overflow/General dev.	58.44 (4.58)	65.39 (3.70)	72.40 (2.87)

ment questions could identify duplicate questions about general software development on Stack Overflow with a decrease of only a couple percent points when compared to the one trained on that data. Models trained on Stack Overflow data also showed similar performance to the one trained on Stack Exchange when detecting duplicate questions on the Game Development Stack Exchange.

We observed the largest decreases in performance when using other models to detect duplicate questions about game development on Stack Overflow. Even then, the decrease in performance was at most three percent points when using the clasifier trained on the Game Development Stack Exchange, which is less than a 10% decrease. Therefore, supervised models trained on other datasets achieved higher performance than the unsupervised and pre-trained models we explored in RQ1, being another viable option for websites with no labelled data available.

The datasets contain several unlabelled duplicate question pairs. Table 3.9 shows a summary describing the duplicate questions pairs that were not ranked among the 20 most similar pairs by the models trained using 1,500 candidate pairs. Between 19% and 25% of the misclassified duplicate pairs did not have their main

questions in the set of candidate questions and therefore our classifier had no chance of correctly ranking those pairs. This loss is justified by the increased performance we obtained by reducing the set of questions the model needed to evaluate. Furthermore, including a larger number of candidates in the evaluation does not necessarily increase the performance of the classifier, as we have shown above.

We also found that many of the misclassified duplicate questions about game development actually had an unlabelled duplicate pair as the top ranked question in the list of most similar pairs. This finding is similar to those found by Zhang et al. [144] and Ahasanuzzaman et al. [3], and stresses the importance of these automatic systems for duplicate question detection. If we considered these unlabelled pairs as correct classifications, the performance of our models could be increased by up to 50%. The percentage of unlabelled duplicates could be even higher if we considered other questions with high ranks, and not just the most similar one.

Finally, we noticed that several main questions discuss more general topics that encompass the specific issue discussed in the duplicate question. For example, question number `116755`[21] on the Game Development Stack Exchange asks about copyright issues with reproducing the mechanics of a specific board game called Risk. That question was marked as a duplicate of another one that discusses how closely a game can resemble another one in general terms[22]. However, our classifier model found a question that is more similar to the first one, as it also discusses copyright issues in creating a reproduction of Risk[23]. Therefore, some of the question pairs marked as duplicates offer additional challenges for automatic detection, as they discuss similar yet different topics and require an understanding of how these topics relate to one another.

[21]https://gamedev.stackexchange.com/questions/116755, accessed September 6th, 2021.
[22]https://gamedev.stackexchange.com/questions/1653, accessed September 6th, 2021.
[23]https://gamedev.stackexchange.com/questions/69119, accessed September 6th, 2021.

Table 3.9: Summary of the duplicate pairs that our supervised models ranked below the 20 most similar pairs. Percentages are shown in relation to the total number of misclassified duplicates in each dataset.

Description	Game Dev. Stack Exchange	Game Dev. Stack Overflow	General Dev. Stack Overflow
Duplicate pairs in test set	221	202	221
Misclassified duplicate pairs	91 (100%)	101 (100%)	53 (100%)
Main question not in the list of candidates	19 (21%)	19 (19%)	13 (25%)
Top ranked question is an unlabelled duplicate	44 (48%)	51 (50%)	10 (19%)
Main question discusses a more general topic	40 (44%)	42 (42%)	28 (53%)

Summary: We could almost double the performance of unsupervised techniques using supervised models trained with labelled data. We obtained the best performance by choosing a number of candidate question pairs in the range of 500 to 2,500. The supervised models could detect duplicate questions on datasets other than the ones they were trained on with a decrease in performance of up to 10%.

3.6 Comparison with other studies

Performing a fair comparison between methodologies for detecting duplicate questions on Stack Overflow is hard as most studies [3, 121, 122, 138, 139, 140, 144] use different datasets and do not provide any code for reproducing their results. For example, several studies [3, 111, 121, 122, 138, 139, 140, 144] have used data collected from a recent Stack Exchange data dump at the time of their writing and sampled it based on post dates and tags to obtain a subset of questions. As the Stack Exchange data dump is mutable (e.g., questions can be edited and deleted after they are posted) and sampling techniques depend on several different parameters, it is very difficult to reproduce and obtain the same dataset.

These issues are also discussed in other studies that have tried to reproduce and compare duplicate detection techniques on Stack Overflow [3, 111, 139]. Silva et al. [111] found a large performance decrease when reproducing the methodology proposed by Ahasanuzzaman et al. [3] and Zhang et al. [144], while also showing that the performance varies greatly when using sets of questions posted in different years. Zhang et al. [139] also tried to reproduce Ahasanuzzaman et al.'s [3] methodology and found a slight increase in performance when evaluating their implementation on their data. All of the studies note the challenges of correctly reproducing those results, as neither the code nor the data are available for the reproduced studies.

We also noticed that several of the studies show at least one design choice that harms the reproducibility of their methodology or its ability to be applied on real-world scenarios. Some of these choices can also artificially boost the performance of the duplicate detection techniques, and the results reported by those studies should

suffer large decreases when applied to real Q&A websites. These pitfalls make the task of performing a fair comparison between studies even harder, as each methodology can use a different approach for evaluating their proposed techniques despite using the same recall-rate measures. We outline below some of the common pitfalls we observed when developing our methodology for duplicate question detection and analyzing those proposed by other studies. We hope that future researchers take these pitfalls into consideration when designing their own systems for duplicate detection, which will help in their adoption by Q&A websites.

1. **Undersampling the test set** - Five of the studies we analyzed [121, 122, 138, 139, 140] have randomly sampled questions that are not part of any duplicate pairs to make the dataset balanced between duplicate and non-duplicate pairs. While this is a valid approach for building a train set, undersampling should not be performed on the test set. The problem of identifying duplicate questions is imbalanced by nature and removing this imbalance during evaluation makes the task easier and not consistent with real-world situations.

2. **Splitting all questions between train and test sets** - Splitting datasets between train and test sets is a common approach for evaluating machine learning techniques. However, when evaluating techniques for duplicate question detection, only the set of duplicate questions should be split between train and test. The remaining questions should be used by both sets, as duplicate questions should be compared to all other answered questions present in the website. Assigning a limited number of questions to be compared in the test set makes the problem easier in a similar manner as undersampling the test set does. We found five studies that split all of the dataset between train and test sets, limiting the number of questions used during evaluation [121, 122, 138, 139, 140].

3. **Appending a '[duplicate]' tag to question titles** - On Stack Overflow and

other Q&A websites of the Stack Exchange network duplicate questions have the tag '[duplicate]' appended to their titles after they are marked as such. While this tag becomes part of the title of the question and can be used for identifying future duplicate relations, it was not present when the question was first posted. Therefore, appending the tag to duplicate questions leads to information leakage and is not representative of real-world scenarios. Four of the studies we analyzed have artificially appended the tag to the titles of duplicate questions [3, 111, 121, 122].

4. **Removing duplicate questions without answers** - Duplicate questions can only point to other questions that already have an answer. However, the duplicate questions themselves do not need to have answers to be marked as such, and duplicates without answers should also be considered in the analysis. We found three studies that removed duplicates without answers [138, 139, 140].

5. **Only comparing duplicates with historical questions** - Ideally, duplicate questions should be identified at the time of their posting, and thus should only point to questions that have been created before that. However, there are several instances of questions that are marked as duplicates of more recent ones. Some examples are the questions `3390396`[24], `1139762`[25], and `5137497`[26], which point to questions that have been posted one to nine months later. Two studies only compared duplicate questions to those posted earlier [3, 144].

6. **Using old data** - We found two studies that used data that was over six years-old at the time of their writing [121, 122]. Using old data may lead to results that are not useful or representative of current Q&A websites. For example, the number of duplicate questions on Stack Overflow grows quickly with the passage of time [3, 111] and the rules for marking questions as duplicates and the way

[24]https://stackoverflow.com/questions/3390396, accessed September 6th, 2021.
[25]https://stackoverflow.com/questions/1139762, accessed September 6th, 2021.
[26]https://stackoverflow.com/questions/5137497, accessed September 6th, 2021.

users interact with them may change. Silva et al. [111] have also shown that the performance of duplicate detection techniques degrades when considering questions posted in more recent years. While publicly available data is scarce for some Q&A websites, the Stack Exchange data dump is updated monthly and researchers should try to use the most recent snapshot available. We note that old data can still be useful as a benchmark for models, but the results obtained when using it should not be expected to hold for current applications.

7. **Training on test duplicate questions** - As we compare duplicate questions with every other answered question in our dataset, it is easy to form training pairs which contain duplicate questions that are also in the test set. These test duplicate questions should therefore be removed from the dataset prior to forming question pairs, to avoid leaking information by using them during training. It is hard to identify if this type of leakage has occurred in other studies, but we have succumbed to this pitfall ourselves during the initial stages of our study before correcting it. Some common signs that the duplicate detection model has been trained on test data are the inability to generalize to other datasets, and high and near-constant values of recall-rates across multiple values of k.

Because of these pitfalls, we chose not to perform a direct comparison between the performance of our techniques and those of other studies. Even if we tried to reproduce other methodologies, the uncertainty of the correctness of our implementations would make the comparison meaningless and unfair. Instead, we have made the datasets we used in our study, together with the implementation of our approach, publicly available in our replication package[27], and we invite other researchers to use these datasets as means of comparing their methodologies to ours. Our datasets have pre-defined train and test splits and can be recreated from the original Stack Exchange data dump using the code found in the package.

[27]Our replication package is available online at https://github.com/asgaardlab/done-21-arthur-duplicate_gamedev_questions-code.

3.7 Implications of our findings

In this section we discuss some of the implications of our findings. We focus our discussion on the implications for the developers of Q&A websites and for researchers, as those are the two main groups that can benefit from our findings.

3.7.1 For the developers of Q&A websites for game development

As we discussed in Section 3.2, Q&A websites suffer with the presence of duplicate questions and a lot of work goes into manually identifying them. Several researchers have tried to help those websites by creating techniques for automatic duplicate detection. Most of those techniques are supervised and require a labelled set of training data that is not available for most Q&A websites for game development. Therefore, in our work, we have explored alternatives for these supervised techniques and have thus analyzed how unsupervised and pre-trained techniques perform in the task of duplicate question detection in those websites.

Our findings show that some of the techniques we analyzed are viable options for the websites that lack the data needed for training supervised models. For example, we could correctly identify up to 39% of duplicate question pairs using a list of the five most similar pairs according to the MPNet similarity. If the Q&A websites implemented that technique, that would lead to a reduction of up to 39% in the posting of questions that have already been answered. We have also shown that other simpler techniques are good choices for unsupervised duplicate detection systems if performance and processing time need to be prioritized.

Furthermore, our results from RQ2 (Section 3.5.2) show that even small sets of labelled data can improve the results provided by the techniques we analyzed in RQ1 (Section 3.5.1). Our supervised models achieved an increased performance in all datasets we used, and almost doubled the performance of detecting duplicate game development questions on Stack Overflow. We trained these models using only a

few hundred pairs of labelled duplicate questions, which is a really small number when compared to the hundreds of thousands of pairs in other websites such as Stack Overflow[28]. Therefore, even small websites can take advantage of their data to improve their duplicate detection systems.

Meanwhile, websites with no labelled data at all can invest some effort into manually labelling a set of a few hundred duplicate pairs to train those supervised models. Another viable option is to train supervised models using the data from other websites (such as Stack Overflow) to create their own duplicate detection systems. As we have shown in RQ2, cross-website models could achieve higher performance than the unsupervised techniques and had only a small decrease in performance when classifying data from a dataset other than the one used for training it.

All of these findings describe viable approaches for developing systems for duplicate question detection. Although the techniques we described may not achieve performance as high as some other tailor-made techniques, they may be the only alternatives for Q&A websites with low resources and no labelled data, such as those focused on discussing game development. Those websites can also use off-the-shelf tools such as ElasticSearch[29] that can be quickly deployed and scaled to help in implementing the techniques. Ultimately, our results can help those websites in building better systems for duplicate detection, which can improve their ability of helping users acquire the information they need.

3.7.2 For researchers

Throughout this chapter, we have discussed several methodological choices that led to the best results for the models and techniques we have studied. For example, we have shown that the performance of the models is affected by the number of candidate

[28]The current number of duplicate question pairs on Stack Overflow is available by running the following query on the Stack Exchange data explorer website https://data.stackexchange.com/stackoverflow/query/1440749/number-of-duplicate-questions-on-stack-overflow.

[29]https://www.elastic.co/elasticsearch/, accessed September 6th, 2021.

question pairs chosen for evaluation and by the parts of the questions that are used for the comparison. We have also analyzed several techniques for comparing questions, and how they perform in the task of ranking duplicate questions and selecting candidate questions. Additionally, we have introduced new methodologies for detecting duplicate questions in the software engineering domain, such as using MPNet and BERTOverflow[30], and using answers from main questions. Future researchers can build upon all of these findings, and use them to decide what are the best approaches to use in their own methodologies.

Moreover, we have tried our best to reproduce a real-world scenario for duplicate detection in our methodology. For example, we compared duplicate questions against answered questions using only the information provided at the time of their posting. We have not altered the contents of the questions aside from preprocessing their texts, and we used all of the questions available in our datasets for evaluating our methodology, without reducing the number of non-duplicate questions in the test set. These choices are different from the ones taken in other studies and can lead to reduced performances when evaluating the methodologies. However, we believe these approaches can better gauge the performance of the proposed techniques when applied on Q&A websites, and should be adopted by other researchers when conducting similar studies into duplicate question detection. Future researchers can therefore use and improve our proposed methodology to build systems that can more closely reflect real-world situations. We have also outlined some of the common pitfalls that occur when evaluating duplicate detection systems in Section 3.6, which can help researchers avoid them in the future.

We have made all of the code and data used in our study available in our replication package. We included thorough explanations and comments so that other researchers can use, reproduce and evaluate our results and methodology. We also made available

[30]Despite being trained using Stack Overflow data, BERTOverflow was created for code and named entity recognition and not was not previously used for duplicate question detection.

all of the models we used (both unsupervised and supervised) so that other researchers can use them without having the burden of retraining from scratch. Everything is bundled in a Docker container to allow running the whole methodology with minimum effort, even on other datasets. Finally, we have introduced fixed datasets for duplicate question detection in the software engineering domain (Section 3.6) to allow for a fair comparison among methodologies. With these measures, we hope to reduce the burden required for reproducing our results and allow for future studies to use and build upon our methodologies and develop better duplicate detection systems.

3.8 Threats to validity

In this section we discuss the threats to the validity of our study.

3.8.1 Internal validity

Throughout this study we performed several preprocessing steps to manipulate our data and adapt it to our needs. While we have not changed any of the content of the posts, our results might have been affected by one of these data processing steps.

We sampled our data from Stack Overflow to obtain a set of game development questions and a small set of general development questions. Despite repeating this sample multiple times using different random seeds, we cannot guarantee that this data is still representative of the whole set of questions from Stack Overflow without further analysis of the remaining data.

We selected only one answer from each question to use when comparing the questions in Section 3.4.3. We used the number of votes, the time of posting, and the flag indicating if the answer was accepted to elect the answer for the comparison. However, it is hard to decide which is the best answer for a given question, and our heuristic for choosing those answers may not lead to the best results. For example, Omondiagbe et al. [80] found that the number of votes and is not a good indicator for answer acceptability and that accepted answers are usually not the first ones to

be posted for questions about Java and JavaScript on Stack Overflow. Future studies should explore using different heuristics for choosing answers for duplicate question detection.

Our results are dependent on our parameter and implementation choices for some of our models and algorithms, such as TF-IDF, BM25, LDA, and Doc2Vec, and other sets of parameters or implementations might offer different results. We also used parameters that were previously tested on Stack Overflow, which may not be the best ones to use for the game development domain. Moreover, we trained our supervised models using a random forest algorithm with parameters defined using a random search approach, and other algorithms with different parameter choices might lead to a better performance. Future studies should explore which sets of parameters, implementations, and algorithms offer the best results for the game development domain.

3.8.2 External validity

In this study we focused our analysis on the data obtained from two Q&A websites, namely the Game Development Stack Exchange and Stack Overflow. We chose to study these websites as they are the two Q&A largest websites for game development that offer a labelled set of duplicate questions. Despite showing that our models can maintain their performance when detecting duplicate questions on different datasets (Section 3.5.2) we cannot guarantee that the models will work on other Q&A websites, whether they are focused on game development or not. Moreover, our results show that the performance of the techniques we tested can vary according to the dataset and using them on other Q&A websites can provide different results than the ones we obtained. Further studies should test the techniques and methodologies we presented on different Q&A websites.

3.8.3 Construct validity

In our study, we ranked question pairs to identify if they are duplicates or not using a set of different similarity scores. We used the recall-rate@k metric to evaluate how we could use these techniques in Q&A websites, simulating a real-world scenario in which we provide users with a list of suggestions. This metric should be more suitable for evaluating the performance in this task than other metrics commonly used for classification as it only considers the samples with the highest scores as opposed to all of them. For example, we could obtain near perfect ROC-AUC (area under the receiver operating characteristic curve) measures of above 0.95 for some similarity scores, as most of the duplicate question pairs were ranked among the top 5% most similar pairs (Section 3.5.1). We note, however, that the number k used for the recall-rates evaluation should be relatively small in order to provide a reasonably-sized list of suggestions.

We did not evaluate whether the techniques can identify that a question does not have a duplicate. Ideally, a real system for detecting duplicate questions should avoid giving wrong recommendations if it is the first time that a question has been posted. Future studies should test whether the techniques are able to correctly detect if a question does not have any duplicates.

Our evaluation is not time aware and the performance of some similarity measures and the supervised classifiers can degrade over time. For example, the vocabulary used in the studied websites can change as new topics and technologies emerge, introducing unseen words and phrases. The unseen text is not a problem for the similarity measures based on MPNet, BERTOverflow, and Jaccard, but may reduce the performance of the measures based on the other techniques that use the vocabulary for training. As a consequence, the supervised classifiers can also suffer a decrease in performance, despite not being directly trained using the vocabulary from the datasets. Future studies should evaluate the results of the techniques and classifiers over time.

3.9 Conclusion

In this chapter, we explored different approaches to identifying duplicate questions on game development Q&A websites. Given that there is a lack of labelled data for duplicate questions about game development, we evaluated seven different unsupervised and pre-trained techniques for this task, including two new techniques which have not been previously used in the software engineering domain. We further improved our results by training supervised models with the small number of labelled duplicate questions about game development. Our main findings include:

(1) Unsupervised and pre-trained techniques could identify up to 54% of the duplicate question pairs about game development among the 20 most similar question pairs. Comparing question titles, bodies, tags, and answers with MPNet offered the best performance.

(2) Supervised models trained on a small set of labelled duplicate questions could almost double the performance obtained by the unsupervised and pre-trained techniques.

(3) Supervised models could predict duplicate questions on datasets other than the ones they were trained on with little to no decrease in performance.

Our results provide valuable insights into the development of systems for duplicate question detection. Furthermore, we have shown that using unsupervised techniques or labelled data from other websites are viable approaches for building a duplicate detection system, which opens new paths for websites with low resources. Ultimately, our findings can be used by developers of those Q&A websites and future researchers to develop systems that can detect duplicate questions more reliably.

Chapter 4

Conclusion & Future Work

4.1 Conclusion

Several studies have analyzed Q&A websites and their many aspects. While most studies focused on general purpose or programming Q&A websites such as Quora and Stack Overflow, none had explored Q&A websites focused on discussing game development and there was little knowledge about how game developers used and interacted with them. In this thesis, we conducted two studies to analyze Q&A websites for game developers from different perspectives and gauge their effectiveness as learning resources and provide insights on how they can improve to provide a better Q&A community for their users. In the first study, we tested the hypothesis that those Q&A websites are useful resources for learning game development skills, but found that the studied websites have decreased their effectiveness and are not helping game developers as much as they used to. In the second study, we tested the hypothesis that we can help the websites by implementing better duplicate detection models with low resources, and found several viable approaches to improving the way they handle duplicate questions. We highlight the main findings and contributions of our studies below.

- In Chapter 2, we analyzed the trends, topics, and user perception of the four largest Q&A websites that discussed game development. We found that the studied Q&A websites are in decline, with the number of posts, users, and

their effectiveness decreasing over time. We also showed that experienced users played a large role in the communities, and that the effectiveness of the communities decreased as they stopped contributing. Moreover, we analyzed the topics discussed by game developers and found that most of them are related to game development, despite the topic about Bug Reports having the largest number of posts among all topics. The topics differ in terms of the length of posts and the frequency in which they discuss code, showing different levels of abstraction and complexity. Finally, users have a mostly negative view of the studied Q&A communities, do not actively participate, and have decrease the frequency with which they access the websites. To help mitigate the decline of these Q&A websites, we created a flowchart with recommendations for game developers to use when deciding where to post their questions.

- In Chapter 3, we tackled the problem of automatically identifying duplicate questions on Q&A websites for game developers. We analyzed the performance of different unsupervised and pre-trained techniques for duplicate question detection, including two new techniques that had not yet been used in the software engineering domain. We showed that comparing all the text contents of questions including their answers with the technique based on MPNet achieved the highest performances without any labelled data. Furthermore, we could gain a large boost in performance when using a small set of labelled data to train supervised models. The supervised models could identify duplicate questions on datasets other than the ones in which they were trained with only a small decrease in performance. Finally, we provide recommendations for developing systems for duplicate question detection, and a replication package that can be used to reproduce and evaluate our results. Future researchers and Q&A websites can used our findings to build better duplicate detection systems, which can lead to better Q&A communities for game developers.

4.2 Future work

The following list presents possible future research directions:

- **Studying other online game development forums and communities** - We focused our study on Q&A websites for game developers. However, as we discussed in Section 2.5.4, game developers also use several other resources for learning and sharing knowledge, such as the official forums and Reddit. Future studies should explore these other resources to identify if they suffer from the same problems as the studied Q&A websites.

- **Analyzing the sentiment of posts on Q&A websites for game developers** - In Chapter 2, we showed that the studied Q&A communities are declining and that users have a negative view towards them. While several factors may be at play, the way users interact with each other may have a large impact on the way the users perceive the community. Future research should analyze the sentiment of questions and answers posted on those Q&A websites to better understand how users interact.

- **Analyzing the code written and shared by game developers** - In our study, we only analyzed text written in natural language on Q&A websites. However, a large portion of the posts we analyzed also contain code snippets that provide valuable information about how game developers write and share code. Future studies should analyze the code provided by those posts.

- **Evaluating the performance of fine-tuned supervised models on Q&A websites for game developers** - In Chapter 3, we analyzed several techniques for duplicate question detection on Q&A websites. We focused on using techniques that required little or no labelled data to provide viable alternatives for websites with a low amount of resources. However, we have shown that our supervised models can also detect duplicate questions on websites other than

those used for training them. Therefore, fine-tuned supervised models trained on other Q&A websites with a large set of labelled data could achieve higher performance than our models. Future studies should evaluate the performance of those fine-tuned models on game development Q&A websites.

Bibliography

[1] D. Abric, O. E. Clark, M. Caminiti, K. Gallaba, and S. McIntosh, "Can duplicate questions on Stack Overflow benefit the software development community?" In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, IEEE, 2019, pp. 230–234.

[2] L. A. Adamic, J. Zhang, E. Bakshy, and M. S. Ackerman, "Knowledge sharing and yahoo answers: Everyone knows something," in *Proceedings of the 17th international conference on World Wide Web*, 2008, pp. 665–674.

[3] M. Ahasanuzzaman, M. Asaduzzaman, C. K. Roy, and K. A. Schneider, "Mining duplicate questions of Stack Overflow," in *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, IEEE, 2016, pp. 402–412.

[4] A. Ahmad, C. Feng, S. Ge, and A. Yousif, "A survey on mining Stack Overflow: Question and answering (Q&A) community," *Data Technologies and Applications*, 2018.

[5] S. Ahmed and M. Bagherzadeh, "What do concurrency developers ask about? A large-scale study using Stack Overflow," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2018, pp. 1–10.

[6] M. Allamanis and C. Sutton, "Why, when, and what: Analyzing Stack Overflow questions by topic, type, and code," in *2013 10th Working Conference on Mining Software Repositories (MSR)*, IEEE, 2013, pp. 53–56.

[7] L. An, O. Mlouki, F. Khomh, and G. Antoniol, "Stack Overflow: A code laundering platform?" In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, IEEE, 2017, pp. 283–293.

[8] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Discovering value from community activity on focused question answering sites: A case study of Stack Overflow," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 850–858.

[9] M. Asaduzzaman, A. S. Mashiyat, C. K. Roy, and K. A. Schneider, "Answering questions about unanswered questions of Stack Overflow," in *2013 10th Working Conference on Mining Software Repositories (MSR)*, IEEE, 2013, pp. 97–100.

[10] K. Bajaj, K. Pattabiraman, and A. Mesbah, "Mining questions asked by web developers," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 112–121.

[11] A. A. Bangash, H. Sahar, S. Chowdhury, A. W. Wong, A. Hindle, and K. Ali, "What do developers know about machine learning: A study of ml discussions on stackoverflow," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, IEEE, 2019, pp. 260–264.

[12] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? An analysis of topics and trends in Stack Overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014.

[13] B. Bazelli, A. Hindle, and E. Stroulia, "On the personality traits of StackOverflow users," in *2013 IEEE international conference on software maintenance*, IEEE, 2013, pp. 460–463.

[14] S. Beyer, C. Macho, M. Di Penta, and M. Pinzger, "What kind of questions do developers ask on Stack Overflow? A comparison of automated approaches to classify posts into question categories," *Empirical Software Engineering*, vol. 25, no. 3, pp. 2258–2301, 2020.

[15] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[16] B. M. Blodgett and A. Salter, "# 1reasonwhy: Game communities and the invisible woman," in *Foundations of Digital Games*, 2014.

[17] B. M. Blodgett and A. Salter, "Hearing 'lady game creators' tweet: #1reasonwhy, women and online discourse in the game development community," in *14th Annual Conference for the Association of Internet Researchers (AoIR)*, 2013.

[18] J. Brookes, M. Warburton, M. Alghadier, M. Mon-Williams, and F. Mushtaq, "Studying human behavior with virtual reality: The Unity experiment framework," *Behavior research methods*, pp. 1–9, 2019.

[19] A. Budhiraja, R. Reddy, and M. Shrivastava, "Lwe: Lda refined word embeddings for duplicate bug report detection," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, 2018, pp. 165–166.

[20] H. Cavusoglu, Z. Li, and K.-W. Huang, "Can gamification motivate voluntary contributions? The case of StackOverflow Q&A community," in *Proceedings of the 18th ACM conference companion on computer supported cooperative work & social computing*, 2015, pp. 171–174.

[21] L. Chen, A. Baird, and D. Straub, "Why do participants continue to contribute? Evaluation of usefulness voting and commenting ootivational affordances within an online knowledge community," *Decision Support Systems*, vol. 118, pp. 21–32, 2019.

[22] T.-H. Chen, S. W. Thomas, and A. E. Hassan, "A survey on the use of topic models when mining software repositories," *Empirical Software Engineering*, vol. 21, no. 5, pp. 1843–1919, 2016.

[23] A. Chowdhury, O. Frieder, D. Grossman, and M. C. McCabe, "Collection statistics for fast duplicate document detection," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 2, pp. 171–191, 2002.

[24] E. Christopoulou and S. Xinogalos, "Overview and comparative analysis of game engines for desktop and mobile devices," *International Journal of Serious Games*, vol. 4, no. 4, 2017.

[25] N. Cliff, "Dominance statistics: Ordinal analyses to answer ordinal questions," *Psychological bulletin*, vol. 114, no. 3, p. 494, 1993.

[26] B. Cowan and B. Kapralos, "A survey of frameworks and game engines for serious game development," in *2014 IEEE 14th International Conference on Advanced Learning Technologies*, IEEE, 2014, pp. 662–664.

[27] D. R. Cox and A. Stuart, "Some quick sign tests for trend in location and dispersion," *Biometrika*, vol. 42, no. 1/2, pp. 80–95, 1955.

[28] V. Cristie and M. Berger, "Game engines for urban exploration: Bridging science narrative for broader participants," in *Playable Cities*, Springer, 2017, pp. 87–107.

[29] A. Cummaudo, R. Vasa, S. Barnett, J. Grundy, and M. Abdelrazek, "Interpreting cloud computer vision pain-points: A mining study of Stack Overflow," *arXiv preprint arXiv:2001.10130*, 2020.

[30] D. H. Dalip, M. A. Gonçalves, M. Cristo, and P. Calado, "Exploiting user feedback to learn to rank answers in Q&A forums: A case study with Stack Overflow," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 2013, pp. 543–552.

[31] S. Deng, J. Tong, Y. Lin, H. Li, and Y. Liu, "Motivating scholars' responses in academic social networking sites: An empirical study on ResearchGate Q&A behavior," *Information Processing & Management*, vol. 56, no. 6, p. 102 082, 2019.

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[33] M. Ellmann, "Same-same but different: On understanding duplicates in Stack Overflow," *Informatik Spektrum*, vol. 42, no. 4, pp. 266–286, 2019.

[34] R. Epp, D. Lin, and C.-P. Bezemer, "An empirical study of trends of popular virtual reality games and their complaints," *IEEE Transactions on Games*, pp. 1–12, 2021.

[35] C. Fang and J. Zhang, "Users' continued participation behavior in social Q&A communities: A motivation perspective," *Computers in Human Behavior*, vol. 92, pp. 87–109, 2019.

[36] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, "Stack Overflow considered harmful? The impact of copy&paste on android application security," in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017, pp. 121–136.

[37] S. J. Fisher and A. Harvey, "Intervention for inclusivity: Gender politics and indie game development," *Loading...*, vol. 7, no. 11, 2013.

[38] D. Ford, K. Lustig, J. Banks, and C. Parnin, ""we don't do that here" how collaborative editing with mentors improves engagement in social Q&A communities," in *Proceedings of the 2018 CHI conference on human factors in computing systems*, 2018, pp. 1–12.

[39] M. Foxman, "United we stand: Platforms, tools and innovation with the Unity game engine," *Social Media+ Society*, vol. 5, no. 4, p. 2 056 305 119 880 177, 2019.

[40] H. Fu and S. Oh, "Quality assessment of answers with user-identified criteria and data-driven features in social Q&A," *Information Processing & Management*, vol. 56, no. 1, pp. 14–28, 2019.

[41] T. Guan, L. Wang, J. Jin, and X. Song, "Knowledge contribution behavior in online Q&A communities: An empirical investigation," *Computers in Human Behavior*, vol. 81, pp. 137–147, 2018.

[42] J. Han, E. Shihab, Z. Wan, S. Deng, and X. Xia, "What do programmers discuss about deep learning frameworks," *Empirical Software Engineering*, vol. 25, no. 4, pp. 2694–2747, 2020.

[43] F. M. Harper, D. Raban, S. Rafaeli, and J. A. Konstan, "Predictors of answer quality in online q&a sites," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2008, pp. 865–874.

[44] R. Harrap, D. Hutchinson, Z. Sala, M. Ondercin, and P. DiFrancesco, "Our GIS is a game engine: Bringing Unity to spatial simulation of rockfalls," in *GeoComputation*, 2019.

[45] A. Harvey, "Becoming gamesworkers: Diversity, higher education, and the future of the game industry," *Television & New Media*, vol. 20, no. 8, pp. 756–766, 2019.

[46] A. Hindle, A. Alipour, and E. Stroulia, "A contextual approach towards more accurate duplicate bug report detection and ranking," *Empirical Software Engineering*, vol. 21, no. 2, pp. 368–410, 2016.

[47] A. Hindle and C. Onuczko, "Preventing duplicate bug reports by continuously querying bug reports," *Empirical Software Engineering*, vol. 24, no. 2, pp. 902–936, 2019.

[48] Y. Homma, S. Sy, and C. Yeh, "Detecting duplicate questions with deep learning," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2016.

[49] Z. Hong, Z. Deng, R. Evans, and H. Wu, "Patient questions and physician responses in a Chinese health Q&A website: Content analysis," *Journal of Medical Internet Research*, vol. 22, no. 4, e13071, 2020.

[50] D. Hoogeveen, A. Bennett, Y. Li, K. M. Verspoor, and T. Baldwin, "Detecting misflagged duplicate questions in community question-answering archives," in *Twelfth international AAAI conference on web and social media*, 2018.

[51] H. Hu, S. Wang, C.-P. Bezemer, and A. E. Hassan, "Studying the consistency of star ratings and reviews of popular free hybrid android and ios apps," *Empirical Software Engineering*, vol. 24, no. 1, pp. 7–32, 2019.

[52] Z. Imtiaz, M. Umer, M. Ahmad, S. Ullah, G. S. Choi, and A. Mehmood, "Duplicate questions pair detection using siamese MaLSTM," *IEEE Access*, vol. 8, pp. 21 932–21 942, 2020.

[53] P. Jaccard, "The distribution of the flora in the alpine zone. 1," *New phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[54] H. Jelodar, Y. Wang, C. Yuan, X. Feng, X. Jiang, Y. Li, and L. Zhao, "Latent dirichlet allocation (lda) and topic modeling: Models, applications, a survey," *Multimedia Tools and Applications*, vol. 78, no. 11, pp. 15 169–15 211, 2019.

[55] J. Jin, Y. Li, X. Zhong, and L. Zhai, "Why users contribute knowledge to online communities: An empirical study of an online social Q&A community," *Information & management*, vol. 52, no. 7, pp. 840–849, 2015.

[56] A. Kamath, S. Gupta, and V. Carvalho, "Reversing gradients in adversarial domain adaptation for question deduplication and textual entailment tasks," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5545–5550.

[57] A. Kamienski and C.-P. Bezemer, "An empirical study of Q&A websites for game developers," *Empirical Software Engineering*, 2021.

[58] C. M. Kanode and H. M. Haddad, "Software engineering challenges in game development," in *2009 Sixth International Conference on Information Technology: New Generations*, IEEE, 2009, pp. 260–265.

[59] I. Koksal, *Video gaming industry & its revenue shift*, Accessed: August 14, 2020, 2019. [Online]. Available: https://www.forbes.com/sites/ilkerkoksal/2019/11/08/video-gaming-industry--its-revenue-shift/#12d74894663e.

[60] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, 2014, pp. 1188–1196.

[61] Z. Li, G. Yin, Y. Yu, T. Wang, and H. Wang, "Detecting duplicate pull-requests in GitHub," in *Proceedings of the 9th Asia-Pacific Symposium on Internetware*, 2017, pp. 1–6.

[62] Z. Li, Y. Yu, M. Zhou, T. Wang, G. Yin, L. Lan, and H. Wang, "Redundancy, context, and preference: An empirical study of duplicate pull requests in OSS projects," *IEEE Transactions on Software Engineering*, 2020.

[63] D. Liang, F. Zhang, W. Zhang, Q. Zhang, J. Fu, M. Peng, T. Gui, and X. Huang, "Adaptive multi-attention network incorporating answer information for duplicate question detection," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 95–104.

[64] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, and M. Lanza, "Pattern-based mining of opinions in Q&A websites," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, IEEE, 2019, pp. 548–559.

[65] M. Linares-Vásquez, G. Bavota, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "How do API changes trigger Stack Overflow discussions? A study on the Android SDK," in *proceedings of the 22nd International Conference on Program Comprehension*, 2014, pp. 83–94.

[66] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk, "An exploratory analysis of mobile development issues using stack overflow," in *2013 10th Working Conference on Mining Software Repositories (MSR)*, IEEE, 2013, pp. 93–96.

[67] D. P. Lopresti, "Models and algorithms for duplicate document detection," in *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR'99 (Cat. No. PR00318)*, IEEE, 1999, pp. 297–300.

[68] S. K. Lukins, N. A. Kraft, and L. H. Etzkorn, "Bug localization using latent dirichlet allocation," *Information and Software Technology*, vol. 52, no. 9, pp. 972–990, 2010.

[69] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design lessons from the fastest Q&A site in the west," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2011, pp. 2857–2866.

[70] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[71] F. Messaoudi, A. Ksentini, G. Simon, and P. Bertin, "Performance analysis of game engines on mobile and fixed devices," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 13, no. 4, pp. 1–28, 2017.

[72] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[73] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[74] Y. Mizobuchi and K. Takayama, "Two improvements to detect duplicates in Stack Overflow," in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, IEEE, 2017, pp. 563–564.

[75] M. R. Morris, J. Teevan, and K. Panovich, "What do people ask their social networks, and why? a survey study of status message q&a behavior," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2010, pp. 1739–1748.

[76] D. Movshovitz-Attias, Y. Movshovitz-Attias, P. Steenkiste, and C. Faloutsos, "Analysis of the reputation system and user contributions on a question answering website: StackOverflow," in *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, IEEE, 2013, pp. 886–893.

[77] S. Nadi and C. Treude, "Essential sentences for navigating Stack Overflow answers," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, IEEE, 2020, pp. 229–239.

[78] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns, "What makes a good code example?: A study of programming Q&A in StackOverflow," in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, IEEE, 2012, pp. 25–34.

[79] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu, "Using of Jaccard coefficient for keywords similarity," in *Proceedings of the international multiconference of engineers and computer scientists*, 2013, pp. 380–384.

[80] O. P. Omondiagbe, S. A. Licorish, and S. G. MacDonell, "Features that predict the acceptability of Java and JavaScript answers on Stack Overflow," in *Proceedings of the Evaluation and Assessment on Software Engineering*, 2019, pp. 101–110.

[81] S. Overflow, *Stack Overflow's 2020 developer survey*, Accessed: August 14, 2020, 2020. [Online]. Available: https://insights.stackoverflow.com/survey/2020.

[82] S. Overflow, *About Stack Overflow*, https://stackoverflow.com/company, Accessed: July 25, 2021, 2021.

[83] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey, "Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow," *Georgia Institute of Technology, Tech. Rep*, vol. 11, 2012.

[84] L. Pascarella, F. Palomba, M. Di Penta, and A. Bacchelli, "How is video game development different from software development in open source?" In *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*, IEEE, 2018, pp. 392–402.

[85] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[86] N. Poerner and H. Schütze, "Multi-view domain adapted sentence embeddings for low-resource unsupervised duplicate question detection," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 1630–1641.

[87] M. F. Porter *et al.*, "An algorithm for suffix stripping.," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[88] D. A. Prabowo and G. B. Herwanto, "Duplicate question detection in question answer website using convolutional neural network," in *2019 5th International Conference on Science and Technology (ICST)*, IEEE, vol. 1, 2019, pp. 1–6.

[89] T. B. Procaci, B. P. Nunes, T. Nurmikko-Fuller, and S. W. Siqueira, "Finding topical experts in question & answer communities," in *2016 IEEE 16th International Conference on Advanced Learning Technologies (ICALT)*, IEEE, 2016, pp. 407–411.

[90] T. B. Procaci, S. W. Siqueira, B. P. Nunes, and T. Nurmikko-Fuller, "Modelling experts behaviour in Q&A communities to predict worthy discussions," in *2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*, IEEE, 2017, pp. 291–295.

[91] M. M. Rahman and C. K. Roy, "An insight into the unresolved questions at Stack Overflow," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, IEEE, 2015, pp. 426–429.

[92] M. S. Rakha, C.-P. Bezemer, and A. E. Hassan, "Revisiting the performance evaluation of automated approaches for the retrieval of duplicate issue reports," *IEEE Transactions on Software Engineering*, vol. 44, no. 12, pp. 1245–1268, 2017.

[93] M. S. Rakha, C.-P. Bezemer, and A. E. Hassan, "Revisiting the performance of automated approaches for the retrieval of duplicate reports in issue tracking systems that perform just-in-time duplicate retrieval," *Empirical Software Engineering*, vol. 23, no. 5, pp. 2597–2621, 2018.

[94] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, Citeseer, vol. 242, 2003, pp. 29–48.

[95] S. Ravi, B. Pang, V. Rastogi, and R. Kumar, "Great question! question quality in community q&a," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 8, 2014.

[96] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, "A large scale study of programming languages and code quality in github," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, pp. 155–165.

[97] R. Řehůřek, *Gensim: Topic modelling for humans*, https://radimrehurek.com/gensim, Accessed: September 5, 2021, 2021.

[98] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019. [Online]. Available: https://arxiv.org/abs/1908.10084.

[99] L. Richardson, *Beautiful soup*, https://www.crummy.com/software/BeautifulSoup, Accessed: September 5, 2021, 2020.

[100] A. Rochette, Y. Yaghoobzadeh, and T. J. Hazen, "Unsupervised domain adaptation of contextual embeddings for low-resource duplicate question detection," *arXiv preprint arXiv:1911.02645*, 2019.

[101] J. Rodrigues, C. Saedi, V. Maraev, J. Silva, and A. Branco, "Ways of asking and replying in duplicate question detection," in *Proceedings of the 6th joint conference on lexical and computational semantics (SEM)*, 2017, pp. 262–270.

[102] J. Romano, J. D. Kromrey, J. Coraggio, J. Skowronek, and L. Devine, "Exploring methods for evaluating group differences on the nsse and other surveys: Are the t-test and cohen'sd indices the most appropriate choices," in *annual meeting of the Southern Association for Institutional Research*, Citeseer, 2006, pp. 1–51.

[103] C. Rosen and E. Shihab, "What are mobile developers asking about? A large scale study using Stack Overflow," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016.

[104] A. Rücklé, N. S. Moosavi, and I. Gurevych, "Neural duplicate question detection without labeled training data," *arXiv preprint arXiv:1911.05594*, 2019.

[105] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," in *29th International Conference on Software Engineering (ICSE'07)*, IEEE, 2007, pp. 499–510.

[106] C. Sadowski, K. T. Stolee, and S. Elbaum, "How developers search for code: A case study," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 191–201.

[107] C. Saedi, J. Rodrigues, J. Silva, A. Branco, and V. Maraev, "Learning profiles in duplicate question detection," in *2017 IEEE international conference on information reuse and integration (IRI)*, IEEE, 2017, pp. 544–550.

[108] T. Santos, K. Burghardt, K. Lerman, and D. Helic, "Can badges foster a more welcoming culture on Q&A boards?" In *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 14, 2020, pp. 969–973.

[109] D. J. Shah, T. Lei, A. Moschitti, S. Romeo, and P. Nakov, "Adversarial domain adaptation for duplicate question detection," *arXiv preprint arXiv:1809.02255*, 2018.

[110] X. Shen, A. L. Jia, S. Shen, and Y. Dou, "Helping the ineloquent farmers: Finding experts for questions with limited text in agricultural Q&A communities," *IEEE Access*, vol. 8, pp. 62 238–62 247, 2020.

[111] R. F. Silva, K. Paixão, and M. de Almeida Maia, "Duplicate question detection in Stack Overflow: A reproducibility study," in *2018 IEEE 25th international conference on software analysis, evolution and reengineering (SANER)*, IEEE, 2018, pp. 572–581.

[112] K. Somasundaram and G. C. Murphy, "Automatic categorization of bug reports using latent dirichlet allocation," in *Proceedings of the 5th India software engineering conference*, 2012, pp. 125–130.

[113] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "MPNet: Masked and permuted pre-training for language understanding," *arXiv preprint arXiv:2004.09297*, 2020.

[114] C. Sun, D. Lo, S.-C. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, IEEE, 2011, pp. 253–262.

[115] C. Sun, D. Lo, X. Wang, J. Jiang, and S.-C. Khoo, "A discriminative model approach for accurate duplicate bug report retrieval," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, 2010, pp. 45–54.

[116] J. Tabassum, M. Maddela, W. Xu, and A. Ritter, "Code and named entity recognition in StackOverflow," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.443/.

[117] U. Technologies, *Unity technologies releases Unity 3.5*, https://unity.com/our-company/newsroom/unity-technologies-releases-unity-3-5, Accessed: June 28, 2021, 2021.

[118] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web? (NIER track)," in *Proceedings of the 33rd international conference on software engineering*, 2011, pp. 804–807.

[119] C. Treude and M. P. Robillard, "Augmenting API documentation with insights from Stack Overflow," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, IEEE, 2016, pp. 392–403.

[120] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[121] L. Wang, L. Zhang, and J. Jiang, "Detecting duplicate questions in Stack Overflow via deep learning approaches," in *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, IEEE, 2019, pp. 506–513.

[122] L. Wang, L. Zhang, and J. Jiang, "Duplicate question detection with deep learning in Stack Overflow," *IEEE Access*, vol. 8, pp. 25 964–25 975, 2020.

[123] Q. Wang, B. Xu, X. Xia, T. Wang, and S. Li, "Duplicate pull request detection: When time matters," in *Proceedings of the 11th Asia-Pacific Symposium on Internetware*, 2019, pp. 1–10.

[124] S. Wang, T.-H. Chen, and A. E. Hassan, "Understanding the factors for fast answers in technical Q&A websites," *Empirical Software Engineering*, vol. 23, no. 3, pp. 1552–1593, 2018.

[125] S. Wang, D. Lo, and L. Jiang, "An empirical study on developer interactions in StackOverflow," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 1019–1024.

[126] Y. Wang, "The price of being polite: Politeness, social status, and their joint impacts on community Q&A efficiency," *Journal of Computational Social Science*, pp. 1–22, 2020.

[127] J. R. Whitson, "Voodoo software and boundary objects in game development: How developers collaborate and conflict with game engines and art tools," *new media & society*, vol. 20, no. 7, pp. 2315–2332, 2018.

[128] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in statistics*, Springer, 1992, pp. 196–202.

[129] W. Witkowski, *Videogames are a bigger industry than movies and north american sports combined, thanks to the pandemic*, https://www.marketwatch.com/story/videogames-are-a-bigger-industry-than-sports-and-movies-combined-thanks-to-the-pandemic-11608654990, Accessed: July 4, 2021, 2020.

[130] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[131] Y. Wu, S. Wang, C.-P. Bezemer, and K. Inoue, "How do developers utilize source code from Stack Overflow?" *Empirical Software Engineering*, vol. 24, no. 2, pp. 637–673, 2019.

[132] B. Xu, T. Hoang, A. Sharma, C. Yang, X. Xia, and D. Lo, "Post2vec: Learning distributed representations of Stack Overflow posts," *IEEE Transactions on Software Engineering*, 2021.

[133] Z. Xu and H. Yuan, "Forum duplicate question detection by domain adaptive semantic matching," *IEEE Access*, vol. 8, pp. 56 029–56 038, 2020.

[134] X.-L. Yang, D. Lo, X. Xia, Z.-Y. Wan, and J.-L. Sun, "What security questions do developers ask? a large-scale study of stack overflow posts," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 910–924, 2016.

[135] M. Zahedi, R. N. Rajapakse, and M. A. Babar, "Mining questions asked about continuous software engineering: A case study of stack overflow," in *Proceedings of the Evaluation and Assessment in Software Engineering*, 2020, pp. 41–50.

[136] H. Zhang, S. Wang, T.-H. Chen, and A. E. Hassan, "Reading answers on stack overflow: Not enough!" *IEEE Transactions on Software Engineering*, 2019.

[137] H. Zhang, S. Wang, T.-H. P. Chen, Y. Zou, and A. E. Hassan, "An empirical study of obsolete answers on Stack Overflow," *IEEE Transactions on Software Engineering*, 2019.

[138] W. E. Zhang, Q. Z. Sheng, J. H. Lau, and E. Abebe, "Detecting duplicate posts in programming QA communities via latent semantics and association rules," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 1221–1229.

[139] W. E. Zhang, Q. Z. Sheng, J. H. Lau, E. Abebe, and W. Ruan, "Duplicate detection in programming question answering communities," *ACM Transactions on Internet Technology (TOIT)*, vol. 18, no. 3, pp. 1–21, 2018.

[140] W. E. Zhang, Q. Z. Sheng, Y. Shu, and V. K. Nguyen, "Feature analysis for duplicate detection in programming QA communities," in *International Conference on Advanced Data Mining and Applications*, Springer, 2017, pp. 623–638.

[141] W. E. Zhang, Q. Z. Sheng, Z. Tang, and W. Ruan, "Related or duplicate: Distinguishing similar CQA questions via convolutional neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 1153–1156.

[142] X. Zhang, S. Liu, X. Chen, *et al.*, "Social capital, motivations, and knowledge sharing intention in health Q&A communities," *Management Decision*, 2017.

[143] Y. Zhang, T. Lu, C. W. Phang, and C. Zhang, "Scientific knowledge communication in online Q&A communities: Linguistic devices as a tool to increase the popularity and perceived professionalism of knowledge contribution," *Journal of the Association for Information Systems*, vol. 20, no. 8, p. 3, 2019.

[144] Y. Zhang, D. Lo, X. Xia, and J.-L. Sun, "Multi-factor duplicate question detection in Stack Overflow," *Journal of Computer Science and Technology*, vol. 30, no. 5, pp. 981–997, 2015.

[145] J. Zhou, S. Wang, C.-P. Bezemer, and A. E. Hassan, "Bounties on technical Q&A sites: A case study of Stack Overflow bounties," *Empirical Software Engineering*, vol. 25, no. 1, pp. 139–177, 2020.

[146] Q. Zhou, X. Liu, and Q. Wang, "Interpretable duplicate question detection models based on attention mechanism," *Information Sciences*, vol. 543, pp. 259–272, 2021.