Noname manuscript No.
(will be inserted by the editor)

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

Arthur Kamienski · Abram Hindle · Cor-Paul Bezemer

Received: date / Accepted: date

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 paper 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 a small set of labelled data to improve the performance 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 Q&A websites for game developers and ultimately providing game developers with a more effective Q&A community.

Keywords Q&A communities · Game development

Arthur Kamienski · Abram Hindle · Cor-Paul Bezemer
University of Alberta
Edmonton, AB, Canada
E-mail: {kamiensk,hindle1,bezemer}@ualberta.ca

1 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 [69]. 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 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 [81], 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 [25] due to factors such as low user interaction and a decrease in the number of answered and resolved questions over time. As answering duplicate questions requires unnecessary effort from answerers, and may cause a delay in getting an answer to a question, it is important to identify such questions so that they can be dealt with appropriately [81]. 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 [2, 58, 65, 66, 76, 77, 81]. While researchers have put a lot of effort into

[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.

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 [2, 65, 66, 76, 77]. Similarly, the performance of these techniques may be affected by the specific characteristics of different communities or different question topics such as game development. Aside from the different programming languages and technologies, Q&A websites for game development also discuss topics that are specific to the game development domain and are not related to software engineering. For example, our previous study [25] showed that game developers discuss topics such as game design, sound, physics, and camera positioning, which do not necessarily involve programming. While some of the knowledge learned in other Q&A websites may transfer well to the game development context, no other studies have analyzed how the techniques perform on game development Q&A websites, either when trained on their own data, or on the data from other domains. Moreover, prior research [63] has shown that Natural Language Processing techniques have difficulties when performing sentiment analysis on game-related text, which supports the need for investigating how they perform in other tasks involving game-related data.

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 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 paper, we analyze how existing pre-trained and unsupervised techniques can be used to detect duplicate game development questions and provide a viable approach for Q&A websites that do not have a large number of labelled duplicate 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

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

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 chose techniques with varying levels of complexity and computational cost, as a way providing websites with different options to choose from depending on their needs and resources. We find that computing the similarity between all question elements (i.e., title, body, tags, and answers) using a model based on MPNet [59] 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 for game developers. 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 for Q&A websites for game developers that have no or a small number of identified duplicate questions;
- 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 paper, allowing researchers to use, reproduce, and evaluate our results in future studies.

Paper outline: The remainder of this paper is organized as follows: Section 2 discusses background and related work. Section 3 describes our methodology and we present our findings in Section 4. In Section 5 we discuss the matter of comparing our methodology to those of other studies, and in Section 6 we discuss the implications of our findings. Section 7 presents the threats to the validity of our study. Finally, Section 8 concludes the paper.

2 Background and related work

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

2.1 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 [37].

Many researchers have previously studied Stack Overflow and its many aspects [3]. For example, Barua et al. [4] have analyzed the topics discussed

[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.

by developers, while Wu et al. [71] have explored how developers utilize the code discussed in it, and Bazelli et al. [5] have explored the personality traits of Stack Overflow users.

Researchers have also analyzed Q&A websites focused on discussing varied topics, such as health [19] and social [14] Q&A websites. Many of these studies searched for ways of improving these websites, such as helping users find information [43, 82] and facilitating their interactions [42, 55, 57, 68], and analyzed users' motivations for participating in the websites [7, 10, 13, 15, 23, 80].

Kamienski and Bezemer [25] 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 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.

2.2 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 [8, 30]. Specifically in the software engineering context, researchers have invested great effort into studying and developing automated detection techniques for duplicate pull requests [27, 28, 67] and bug reports [16, 17, 45, 46].

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 [12] and impacts in the community [1]), the majority of work focused on developing systems to automatically detect duplicate questions. For example, Zhang et al. [81] introduced DupPredictor, which uses title, description, topic and tag similarities to identify duplicates. Building on that, Ahasanuzzaman et al. [2] 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. [76, 77] 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. [65, 66] 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 [20, 33, 58, 72, 78, 79].

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] [18, 21, 41, 51, 54, 83]. Studies have also explored how to use domain adaptation to create duplicate detection techniques with no labelled data [24, 29, 39, 50, 52, 56, 73]. 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 Methodology

In this section we describe our methodology for collecting and processing the data we used in our study, and applying techniques for duplicate question detection. Fig. 1 shows an overview of the steps we have taken in our methodology. The code, data, and models used in this study are available online in our replication package[9] to allow future researchers to use, reproduce, and evaluate our results.

3.1 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

[8] `https://stackexchange.com/`, acessed July 4, 2021.

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

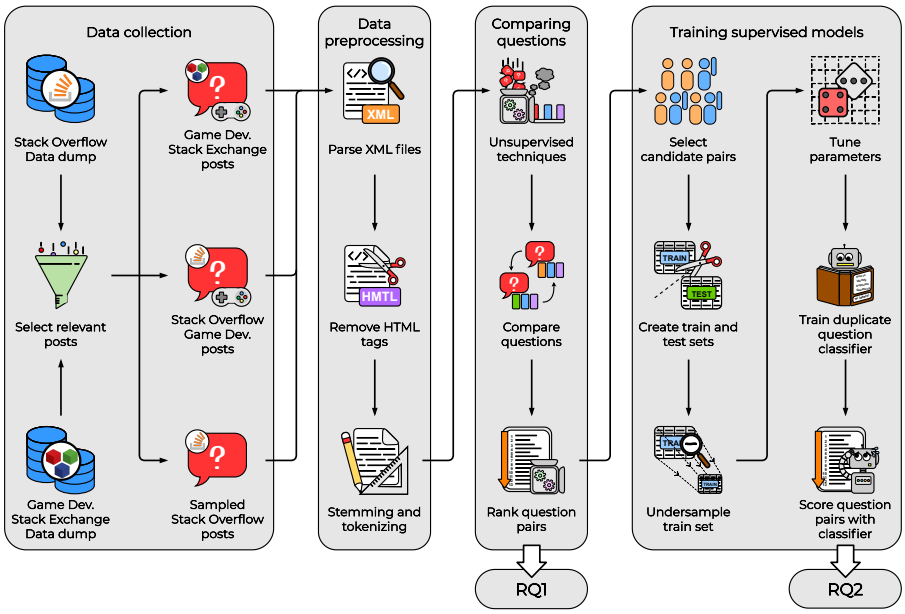

Fig. 1 Overview of the steps we have taken in our methodology.

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 [25] 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-development-kit'*, *'unreal-blueprint'*, *'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

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

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. We performed our following experiments separately for each of those five samples. However, for the remainder of this paper we refer to the five samples as a single dataset (Stack Overflow - General Development) and report the mean and standard deviation obtained for each of our results.

Table 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. We note that the table only shows the number of labelled duplicate questions in our datasets and the true percentage of duplicate questions is potentially much higher in those websites.

Table 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

3.2 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

[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.

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 3.3. If the question had no accepted answers, we chose the one with the largest number of votes that was posted first. Other studies have also used the number of votes received by answers as proxies for their quality [9, 34, 44]. 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 [4, 58, 62, 74]. 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 [49] 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 [84] to remove punctuation, multiple whitespaces, numeric characters, stopwords, and short words. The function also stems the texts using the Porter Stemmer [40] and tokenizes them by splitting words separated by spaces.

3.3 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 [2, 81], to more complex deep-learning-based techniques [65, 66]. 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[12] as they do not demand labelled data and can be implemented with relatively few computational resources[13].

[12] We define pre-trained techniques as any technique that requires a training step prior to being used for our task of detecting duplicate questions. This training step is usually unsupervised and does not use any additional information about our data aside from its vocabulary. While we pre-trained some of the techniques ourselves using the data we described in Section 3.1, we also used techniques that were pre-trained by their authors using a separate set of data.

[13] We performed all of our experiments on a laptop with an 8th generation Intel Core i7 processor and 16GB of RAM. We also used the Tesla P100 GPUs provided by Google Colab

While custom techniques created to perform specific tasks can achieve higher performance on their domains, the cost of implementing them is higher (both in terms of amount of data and computation), and they may not be a feasible alternative for some websites. Nevertheless, Q&A websites can use our results as a starting point to identify which techniques are better suited for their needs, as the increased costs may not always justify the differences in performance.

Table 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 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	[58, 81]
TF-IDF	No	No	Yes	[76, 77]
BM25	No	No	Yes	[2, 58, 77]
Topic	Yes	No	Yes	[58, 76, 77, 81]
Doc2Vec	Yes	No	Yes	[76, 77]
BERTOverflow	Yes	No	No	
MPNet	Yes	Yes	No	

We also analyzed how using different text elements from questions affect the performance of the techniques in Table 2. Fig. 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 [2, 58, 77, 78, 81] 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, 29], and we thus evaluate the impact of their usage in our method-

Pro, which are available for a small monthly fee. Using this setup, the most costly techniques took 11 minutes to run.

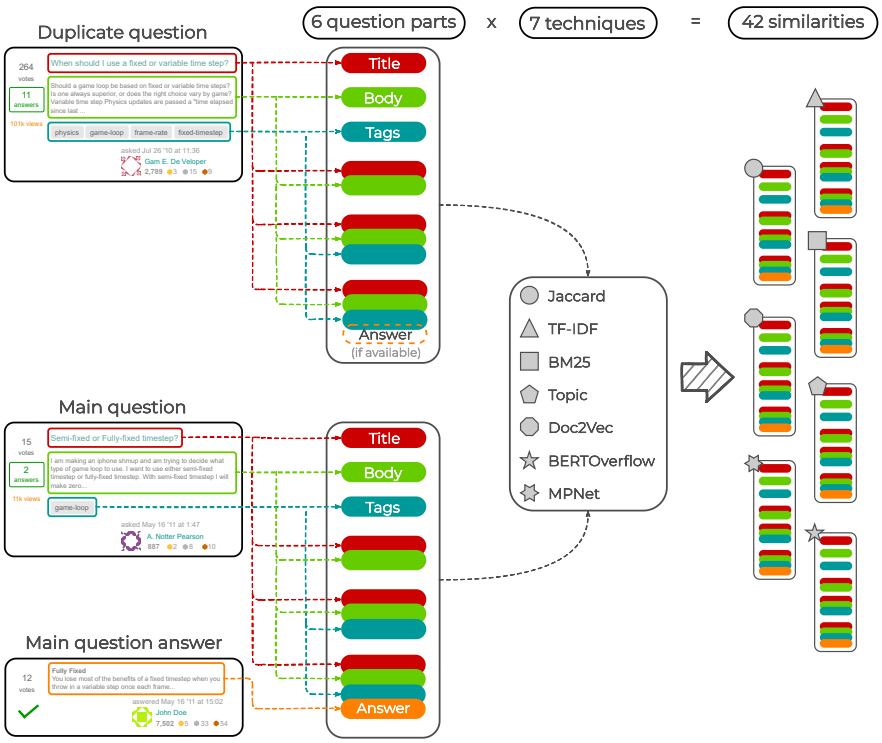

Fig. 2 Overview of the methodology we used for comparing questions using seven techniques.

ology. While answers would not be available for recently posted questions, they are still present for some of the questions that have been previously posted. Therefore, the comparison using answers may still prove useful for detecting duplicates, even when one of the questions do not have an answer.

We applied each technique to each of these documents, obtaining 42 different similarity measures for each question pair (6 documents × 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[14]. We provide further explanations of the comparison techniques we used in the following sections.

3.3.1 Jaccard similarity

The Jaccard similarity coefficient [22] 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 [35]. Before calculating the Jaccard similarity between two texts, we converted them into sets by

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

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.

3.3.2 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 [47]. In our study, we used the `TfidfVectorizer` implementation provided by Python's scikit-learn library [38] 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.

3.3.3 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 [84][15]. Based on the study of BM25 parameters for duplicate question detection on Stack Overflow performed by Ahasanuzzaman et al. [2], we defined the values of the free parameters k_1 and b as 0.05 and 0.03, respectively.

3.3.4 Topic similarity

We measured the similarity between the topics of two documents using the latent Dirichlet allocation (LDA) algorithm [6]. 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

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

each dataset, totalling 18 different models (6 document sets × 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 [77, 78]. 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 [64]. We do not use the cosine similarity for comparing the topic vectors, as they do not represent a point in a multi-dimensional space. Instead, 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 such as the ones we obtained from the LDA algorithm.

3.3.5 Doc2Vec similarity

Doc2Vec [26] is an unsupervised algorithm based on Word2Vec [31, 32] 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 × 3 datasets). We used the same parameters as indicated by Zhang et al. [76], which are shown in Table 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 Parameters used for learning document embeddings using Gensim's Doc2Vec implementation, following suggestions from Zhang et al. [76].

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

3.3.6 BERTOverflow similarity

BERTOverflow [62] 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 [11], 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[16], 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 [70][17] with default parameters, and adapted it to produce document vectors using the SentenceTransformers [48][18] package. We performed no other training steps to tune 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.

3.3.7 MPNet similarity

MPNet [59] is another deep neural network-based model for creating word embeddings for natural language processing tasks. It was pre-trained by the original authors using over 160GB of data and 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 [48] package to produce document vectors. This model is based on MPNet and its authors fine-tuned it for the task of producing document vectors on several datasets from different sources which include Q&A websites (but not the ones used in our study). This model currently offers the best average performance on a set of document comparison tasks[19]. 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 Training supervised classifier models

The question comparison techniques we described in Section 3.3 provide similarity scores that help us in identifying duplicate questions. However, each

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

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

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

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

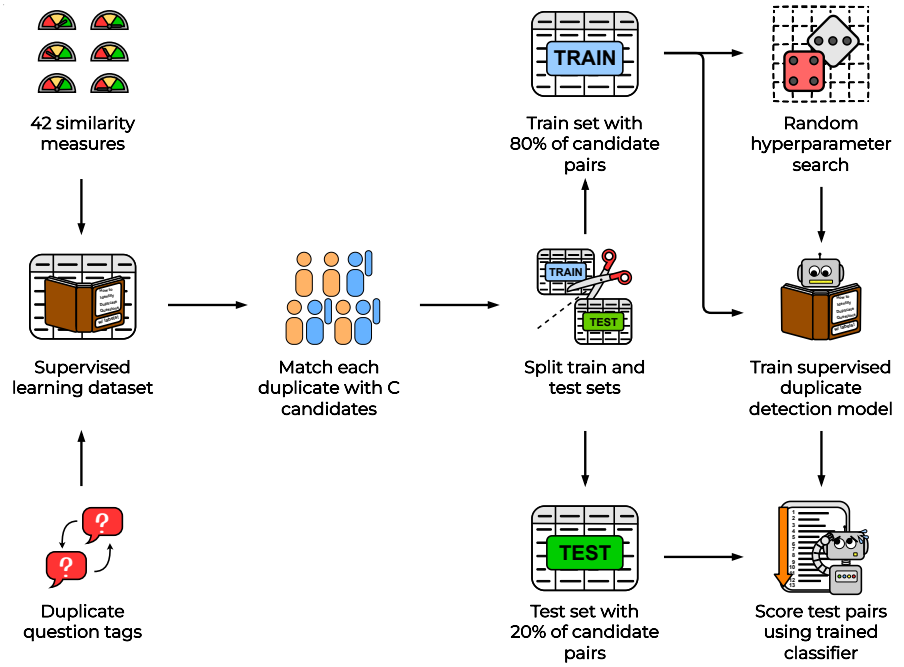

Fig. 3 Overview of the steps we have taken for training supervised models based on the other similarity measures.

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 questions.

Fig. 3 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. [77]. We calculate the new measure by first applying the techniques described in Section 3.3 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. [2] 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 3.3 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 3.3 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 [65, 66, 76, 77]. Instead, we chose to use the random forest algorithm as it obtained some of the best results in other studies [76, 77] 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 iterations of a random parameter search with 5-fold cross validation. Therefore, we trained 30 different models using parameters randomly selected

from a pre-defined set of parameters. We trained each of those models five times using the training sets, each with a different validation fold comprised of 20% of the data. 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.

4 Results

In this section we discuss the motivation, approach and findings for each of our research questions. Section 4.1 discusses the performance of the similarity scores obtained from the techniques described in Section 3.3, while Section 4.2 analyzes the performance of the supervised classifier models described in Section 3.4.

4.1 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 with low resources as alternatives to more complex techniques that may not be easily implemented. Moreover, our findings may help those websites to decide what the best approach for improving their duplicate detection systems is given their circumstances.

Approach: We used the similarity scores described in Section 3 to create different ranks for all of the question pairs produced from comparing duplicate questions to other answered questions in the dataset. Fig. 4 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.

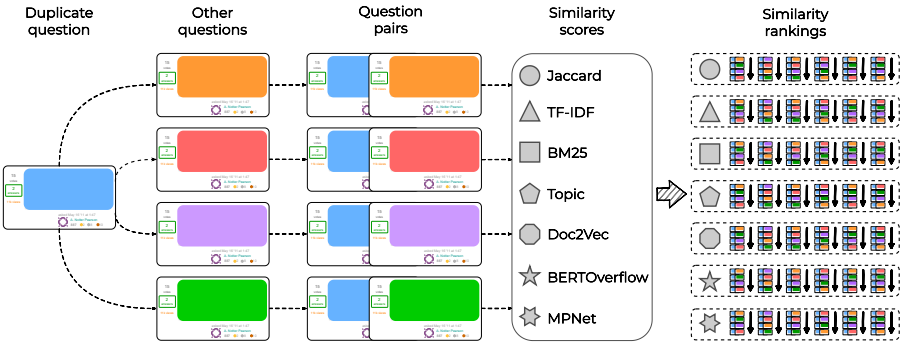

Fig. 4 Overview of our approach to ranking question pairs according to similarity scores.

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 [2, 53, 58, 60, 61, 65, 66, 77, 81], 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. Therefore, this measure can be viewed as a proxy for how likely it is to find a duplicate question 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 4 shows 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 ques-

tions 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. We further investigate the reasons behind those low values found for game development websites in Section 4.2 by performing a manual analysis in the test sets used for our supervised models.

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, respectively. 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[20]. 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 [2, 77, 81].

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

[20] The BM25 similarities scored at most 3 percent points more than MPNet for the game development dataset from Stack Overflow when using answers for the comparison. Despite having a lower performance, we still consider MPNet a more robust approach for detecting duplicates, as it has surpassed the second best approach in every other experiment we performed, with margins of up to 10 percent points.

Table 4 Performance of the studied techniques according to different metrics. 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.

Dataset	Metric	Technique	1	2	3	4	5	6
Game Dev. Stack Exchange	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**
Stack Overflow/ Game dev.	recall-rate@5	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**
		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
	recall-rate@10	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**
		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
	recall-rate@20	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**
		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
Stack Overflow/ General dev.	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)

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 comparing 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`[21] 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`[22], 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 [77, 78, 81]. However, our results differ from the ones found by Zhang et al. [77, 78], 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 5.

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 5 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 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 4, 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

[21] `https://gamedev.stackexchange.com/questions/86517`, accessed September 6th, 2021.

[22] `https://gamedev.stackexchange.com/questions/86326`, accessed September 6th, 2021.

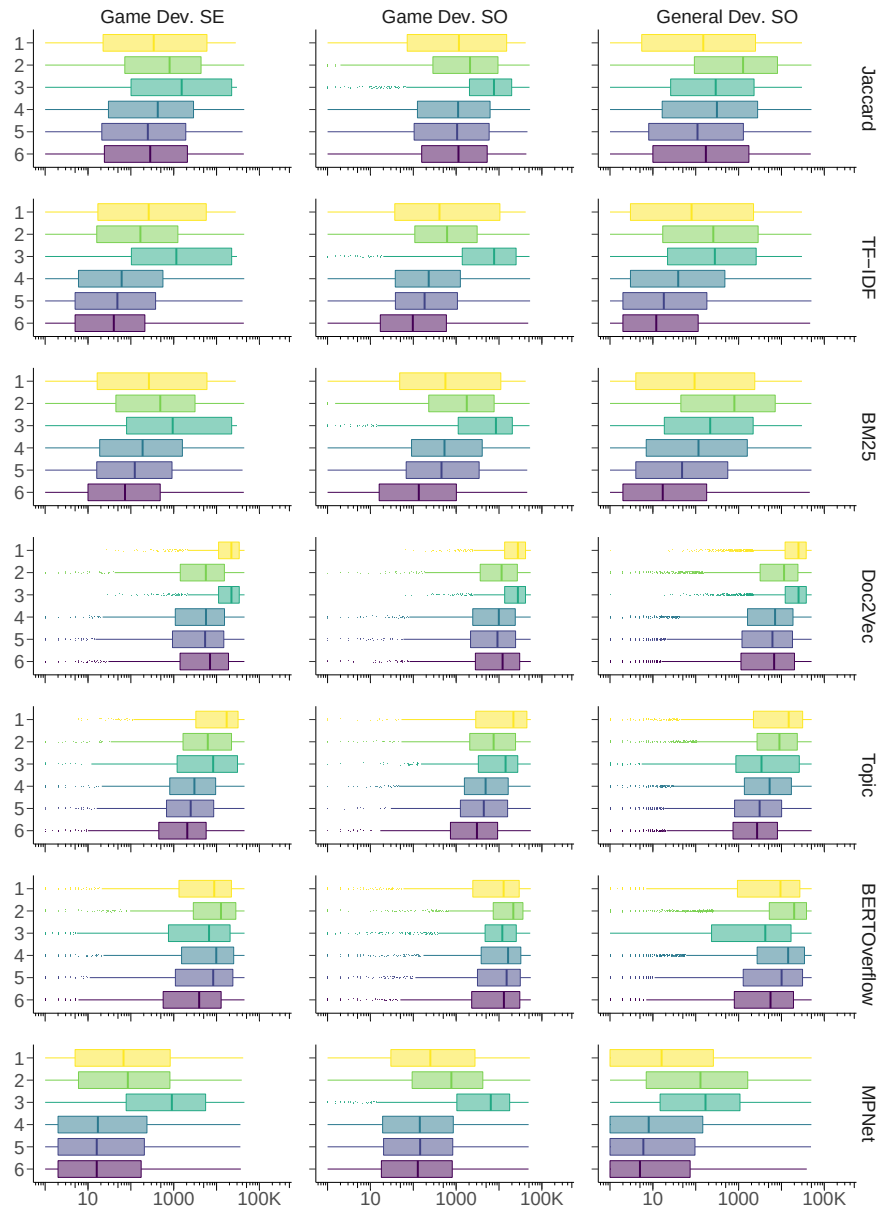

Fig. 5 Distribution of ranks of true duplicate pairs in the three different datasets used in this study according to different techniques. Labels 1 to 5 indicate the question part that was used for comparing the questions, as such: 1 - *title*, 2 - *body*, 3 - *tags*, 4 - *title* and body, 5 - *title*,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

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. 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.

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

Motivation: In RQ1 (Section 4.1) 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 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 a small set of labelled duplicate question data to build a better similarity score that can more reliably detect duplicate questions. We also explore how well models perform when evaluating questions from other websites different than the ones used for training them. Therefore, our findings may help Q&A websites with no or only a small number of labelled duplicate questions by showing how they can leverage a very limited set of labeled data to improve their duplicate detection techniques, even if that data comes from other websites.

Approach: We followed the methodology described in Section 3.4 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 4.1), 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[23]. 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 to create train and test sets. We trained Random Forest models using the hyperparameters that provided the 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.

[23] 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.

Table 5 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)

Findings: **The optimal number of candidates depends on the dataset and metric being used.** Table 5 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 5 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 4.1). 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%.

We observed an increase in performance even when removing the features based on MPNet and BERTOverflow from our classifiers. Without those features, the classifier trained on game development questions from Stack Overflow using 1,500 candidates achieved a recall-rate@5 of 27.23 (69% increase when compared to the best similarity scores in Section 4.1) and a recall-rate@20 of 45.54 (a 36% increase). The classifiers trained using the other datasets also showed significant gains (from 15% to 34%) when compared to the similarity scores used for training them after we removed the features, but could not beat the best similarity scores we obtained from our unsupervised methods. For example, the classifier trained on general development questions on Stack Overflow could only match the previous scores obtained from MPNet, while the one trained on the Game Development Exchange saw a slight decrease of 1 to 4 percent points. Overall, removing features based on MPNet and BERTOverflow caused a decrease of 5% to 15% in the performance of the classifiers.

The models could predict duplicate questions on other datasets with little to no decrease in performance. Table 6 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 development 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.

Table 6 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)

The datasets contain several unlabelled duplicate question pairs. Table 7 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. [81] and Ahasanuzzaman et al. [2], 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`[24] 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[25]. 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[26]. Therefore, some of the question pairs marked as duplicates offer additional

[24] `https://gamedev.stackexchange.com/questions/116755`, accessed September 6th, 2021.

[25] `https://gamedev.stackexchange.com/questions/1653`, September 6th, 2021.

[26] `https://gamedev.stackexchange.com/questions/69119`, accessed September 6th, 2021.

Table 7 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%)

challenges for automatic detection, as they discuss similar yet different topics and require an understanding of how these topics relate to one another.

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%.

5 Comparison with other studies

Performing a fair comparison between methodologies for detecting duplicate questions on Stack Overflow is hard as most studies [2, 65, 66, 76, 77, 78, 81] use different datasets and do not provide any code for reproducing their results. For example, several studies [2, 58, 65, 66, 76, 77, 78, 81] 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 [2, 58, 77]. Silva et al. [58] found a large performance decrease when reproducing the methodology proposed by Ahasanuzzaman et al. [2] and Zhang et al. [81], while also showing that the performance varies greatly when using sets of

questions posted in different years. Zhang et al. [77] also tried to reproduce Ahasanuzzaman et al.'s [2] 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 [65, 66, 76, 77, 78] 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 [65, 66, 76, 77, 78].
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 [2, 58, 65, 66].

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 [76, 77, 78].
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`[27], `1139762`[28], and `5137497`[29], which point to questions that have been posted one to nine months later. Two studies only compared duplicate questions to those posted earlier [2, 81].
6. **Using old data** - We found two studies that used data that was over six years-old at the time of their writing [65, 66]. 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 [2, 58] and the rules for marking questions as duplicates and the way users interact with them may change. Silva et al. [58] 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, it is very hard to compare results across studies, as they might not reflect the true performance of a duplicate detection system

[27] `https://stackoverflow.com/questions/3390396`, accessed September 6th, 2021.
[28] `https://stackoverflow.com/questions/1139762`, accessed September 6th, 2021.
[29] `https://stackoverflow.com/questions/5137497`, accessed September 6th, 2021.

in a real-world scenario. Even if we tried to reproduce other methodologies, the uncertainty of the correctness of our implementations would make the comparison meaningless and unfair. Furthermore, we would need to remedy the pitfalls we mention in order to perform a fair evaluation, which would significantly alter the original approaches. That way, we would not be comparing our technique to a proven baseline, but to a new and untested approach. That is why we tested and compared several techniques that have already been used in other studies [2, 58, 76, 77] (e.g., TF-IDF and BM25) and used them as baselines instead.

Nevertheless, we decided to perform a comparison against the approach described by Silva et al. [58], as it is the only study whose results can be reproduced. To perform the comparison, we evaluated our approach on data extracted from the MSR '15 mining challenge dataset [75][30], which is similar to the one used by Silva et al.[58] and Ahasanuzzaman et al. [2]. The data is comprised of questions marked with the tags *Java*, *C++*, *Python*, *Ruby*, *HTML*, and *Objective-C* collected from the September 2014 Stack Overflow data dump. We judged this dataset to be a suitable choice for evaluating duplicate detection methodologies as it provides different sets of questions from specific topics of the software engineering domain. These sets vary in size, but all of them have a large yet manageable number of a few hundred thousand questions. Table 8 shows a summary of each of the sets of questions contained in the dataset.

Table 8 Summary of the dataset we used for comparing our methodology with other studies that analyzed duplicate questions on Stack Overflow.

Tag	Questions	Non-duplicates	Duplicates	Pairs
Java	708,473	694,330	14,143 (2.00%)	15,070
C++	314,869	306,330	8,539 (2.71%)	9,291
Python	339,652	332,117	7,535 (2.22%)	8,032
Ruby	114,414	113,333	1,081 (0.94%)	1,146
HTML	337,341	334,405	2,936 (0.87%)	3,048
Objective-C	200,862	197,755	3,107 (1.55%)	3,287

We took the following steps to create the dataset starting from the original mining challenge data. First, we selected all of the questions that contain one of the tags mentioned above and assigned them to separate sets. A question can belong to more than one set if it contains more than one of those tags. As the list of question tags is represented as a string in the XML files containing the data, we split the string into a list of tags and only selected the tags that were perfect matches in lowercase. This is an important step to avoid false positives such as *JavaScript* or *IPython*. Finally, for each question set, we extracted duplicate question pairs by selecting only the pairs that have both the duplicate and original questions in the set. We cannot guarantee that the

[30] The dataset is available at `http://2015.msrconf.org/challenge.php`, accessed June 1st, 2022.

resulting dataset is the exactly the same as the one used by Ahasanuzzaman et al. [2], as they did not provide a clear methodology for selecting the questions nor reported the number of questions they obtained for each tag. We also did not alter the content of the questions by appending a `[Duplicate]` label to the title of duplicate questions, as was done in their study.

Other studies [58, 65, 66, 77] have tried to approximate this dataset as a way of comparing methodologies with the one proposed by Ahasanuzzaman et al. [2]. The datasets used in those studies are different from the one we present here, as each uses a different methodology for sampling questions from more recent data dumps. Three of those studies [65, 66, 77] have also randomly sampled questions that are not part of any duplicate pairs to make the dataset balanced between duplicate and non-duplicate pairs. For example, for questions with the *Ruby* tag in our dataset, those studies would randomly select 2,292 questions that are not part of the 1,146 duplicate question pairs to produce a dataset of around 4,500 questions. While undersampling is a valid approach for training classification models, it is not suitable for testing and evaluating their performance in a real-world scenario, and therefore we are not able to compare our results to theirs.

We followed the same methodology as described in Section 3 to compare questions, apply the unsupervised and pre-trained techniques and train supervised classifiers for each of the question sets in both datasets. Here we show the results obtained when selecting 1,500 candidates prior to training and evaluating the model. All of the models we used, along with the train and test sets for each question set are also available in our replication package to allow for future researchers to use, reproduce, and evaluate our results.

Table 9 shows the results we obtained in each of these sets using our supervised models. Additionally, we show the results we obtained by calculating the MPNet similarity technique between all question parts and answers, as it was the best technique we found in RQ1 (Section 4.1). We only show the best results obtained by the studies which used some variation of the Stack Overflow 2014 data dump. Therefore, we only included the results reported by Silva et al. [58] for their reproductions of Dupe [2] and DupPredictor [81], as those are the only two other methodologies that used a dataset that was not undersampled. As we can see, our comparison shows that our MPNet model always outperformed DupeRep, but not always DupPredictorRep. However, our supervised model (RQ2) outperformed DupeRep and DupPredictorRep for all studied tags. We stress that even though we did our best to recreate the dataset as much as possible, the results in Table 9 were reported by Silva et al. [58] for their unique datasets and this is not a completely fair comparison.

6 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

Table 9 Comparison of the different approaches proposed by other studies for detecting duplicate questions on data extracted from Stack Overflow up to 2014 without sampling. Values in bold show the best results obtained for each tag.

Tag	Technique	Source	Recall-rate@		
			5	10	20
Java	DupeRep	[58]	19.44	23.90	27.95
	DupPredictorRep-T + tag	[58]	28.04	36.70	44.02
	MPNet	this	26.26	33.01	40.01
	Supervised model	this	**32.06**	**39.23**	**46.09**
C++	DupeRep	[58]	16.26	20.63	25.67
	DupPredictorRep-T + tag	[58]	18.19	25.91	33.73
	MPNet	this	24.59	32.61	39.70
	Supervised model	this	**31.09**	**39.05**	**46.08**
Python	DupeRep	[58]	16.38	20.00	24.72
	DupPredictorRep-T + tag	[58]	0.00	22.61	34.38
	MPNet	this	25.41	32.85	39.88
	Supervised model	this	**31.00**	**38.02**	**45.06**
Ruby	DupeRep	[58]	29.84	35.45	38.26
	DupPredictorRep-T + tag	[58]	17.31	30.99	39.47
	MPNet	this	43.06	52.78	60.65
	Supervised model	this	**53.24**	**59.72**	**66.67**
HTML	DupeRep	[58]	18.15	21.67	25.30
	DupPredictorRep-T + tag	[58]	28.21	35.89	43.56
	MPNet	this	29.64	34.41	40.37
	Supervised model	this	**30.66**	**37.65**	**45.31**
Objective-C	DupeRep	[58]	23.62	30.64	38.10
	DupPredictorRep-T + tag	[58]	29.52	33.85	39.37
	MPNet	this	29.95	38.00	46.70
	Supervised model	this	**39.13**	**46.86**	**54.27**

for researchers, as those are the two main groups that can benefit from our findings.

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

As we discussed in Section 1, 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.

As we have shown in Section 4.1, 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, we evaluated several approaches to building duplicate detection systems, such as using different question parts and evaluation metrics. Therefore, Q&A websites can use our results to decide which technique better suits their needs.

Furthermore, our results from RQ2 (Section 4.2) show that even small sets of labelled data can improve the results provided by the techniques we analyzed in RQ1 (Section 4.1). 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 eight hundred pairs of labelled duplicate questions, which is a really small number when compared to the hundreds of thousands of pairs on other websites such as Stack Overflow[31]. Therefore, even small websites can take advantage of their data to improve their duplicate detection systems.

Meanwhile, websites with few or no labelled data at all can invest some effort into manually labelling a set of eight hundred duplicate pairs to train and improve the performance those supervised models. This approach would work even if only a small portion of true duplicates are manually detected, as we have shown that our models still perform well when using an imperfect ground truth (Section 4.2). 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.

We note that there is an increased cost associated with training and implementing the supervised models we described. However, this cost should be low when compared to the other unsupervised techniques we described in RQ1 (Section 4.1), as the number of training samples is very low. For example, it took less than ten minutes to train the supervised models in a laptop with an 8th generation Intel Core i7 processor. Nevertheless, Q&A websites should use our results to decide if the increase performance justifies the additional cost based on their circumstances.

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[32] that can be quickly deployed and scaled to help in implementing the techniques. Ultimately, our results can help

[31] 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`.

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

those websites in building better systems for duplicate detection, which can improve their ability of helping users acquire the information they need.

We note, however, that even our best approaches wrongly identified a high number of false-positive duplicate questions, as we showed in Section 4.2. While a more powerful classifier could possibly achieve better results, there is still a degree of subjectivity when it comes to defining what is a duplicate question, and the correct label may change according to the person who assigned it. Therefore, it is extremely hard to assert whether two questions are duplicates, and there may not be a bullet-proof solution to solve the problem of finding duplicate questions. Instead, our approaches try to rank questions based on how likely they are to be duplicates. That way, websites can use these duplicate detection systems to aid them in finding and dealing with those questions.

6.2 For researchers

Throughout our paper, 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 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[33], 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

[33] 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.

pitfalls that occur when evaluating duplicate detection systems in Section 5, 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 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 5) 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.

7 Threats to validity

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

7.1 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 3.3. 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. [36] 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.

We identified in Section 4.2 that the datasets we used contain several unlabelled duplicate pairs. As these labels are crucial for training and evaluating the approaches we explored, our results may vary depending on how many wrong labels are present in the dataset. As the duplicate question labelling mechanism is manual on websites such as Stack Overflow (i.e., by users and moderators), such wrong and/or incomplete labels are likely to exist on all technical Q&A websites that label duplicate questions. Future studies should analyze how the presence and proportion of these wrong labels affect the performance of our approaches.

7.2 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. While some transferability is expected between Q&A websites of similar domains and despite showing that our models can maintain their performance when detecting duplicate questions on different datasets (Section 4.2), 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.

7.3 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 4.1). 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.

8 Conclusion

In this paper, 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. Based on our results, we suggest that websites consider evaluating supervised models with MPNet as a feature to see if this combination improves the performance of their duplicate detection systems if they have the resources required to implement this approach, as this combination yielded good results for most of our experiments. 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.

Conflict of interest

The authors declare that they have no conflict of interest.

Data availability

The datasets generated and analysed during the current study are available as part of our replication package on Zenodo on `https://zenodo.org/record/5500268#.YOiVjtLMJhF`.

References

1. Abric, D., Clark, O.E., Caminiti, M., Gallaba, K., McIntosh, S.: Can duplicate questions on Stack Overflow benefit the software development community? In: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), pp. 230–234. IEEE (2019)
2. Ahasanuzzaman, M., Asaduzzaman, M., Roy, C.K., Schneider, K.A.: Mining duplicate questions of Stack Overflow. In: 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR), pp. 402–412. IEEE (2016)
3. Ahmad, A., Feng, C., Ge, S., Yousif, A.: A survey on mining Stack Overflow: Question and answering (Q&A) community. Data Technologies and Applications (2018)
4. Barua, A., Thomas, S.W., Hassan, A.E.: What are developers talking about? An analysis of topics and trends in Stack Overflow. Empirical Software Engineering **19**(3), 619–654 (2014)
5. Bazelli, B., Hindle, A., Stroulia, E.: On the personality traits of StackOverflow users. In: 2013 IEEE international conference on software maintenance, pp. 460–463. IEEE (2013)
6. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research **3**(Jan), 993–1022 (2003)
7. Chen, L., Baird, A., Straub, D.: Why do participants continue to contribute? Evaluation of usefulness voting and commenting motivational affordances within an online knowledge community. Decision Support Systems **118**, 21–32 (2019)
8. Chowdhury, A., Frieder, O., Grossman, D., McCabe, M.C.: Collection statistics for fast duplicate document detection. ACM Transactions on Information Systems (TOIS) **20**(2), 171–191 (2002)

9. Dalip, D.H., Gonçalves, M.A., Cristo, M., Calado, P.: 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, pp. 543–552 (2013)
10. Deng, S., Tong, J., Lin, Y., Li, H., Liu, Y.: Motivating scholars' responses in academic social networking sites: An empirical study on ResearchGate Q&A behavior. Information Processing & Management **56**(6), 102082 (2019)
11. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
12. Ellmann, M.: Same-same but different: On understanding duplicates in Stack Overflow. Informatik Spektrum **42**(4), 266–286 (2019)
13. Fang, C., Zhang, J.: Users' continued participation behavior in social Q&A communities: A motivation perspective. Computers in Human Behavior **92**, 87–109 (2019)
14. Fu, H., Oh, S.: Quality assessment of answers with user-identified criteria and data-driven features in social Q&A. Information Processing & Management **56**(1), 14–28 (2019)
15. Guan, T., Wang, L., Jin, J., Song, X.: Knowledge contribution behavior in online Q&A communities: An empirical investigation. Computers in Human Behavior **81**, 137–147 (2018)
16. Hindle, A., Alipour, A., Stroulia, E.: A contextual approach towards more accurate duplicate bug report detection and ranking. Empirical Software Engineering **21**(2), 368–410 (2016)
17. Hindle, A., Onuczko, C.: Preventing duplicate bug reports by continuously querying bug reports. Empirical Software Engineering **24**(2), 902–936 (2019)
18. Homma, Y., Sy, S., Yeh, C.: Detecting duplicate questions with deep learning. In: Proceedings of the International Conference on Neural Information Processing Systems (NIPS) (2016)
19. Hong, Z., Deng, Z., Evans, R., Wu, H.: Patient questions and physician responses in a Chinese health Q&A website: Content analysis. Journal of Medical Internet Research **22**(4), e13071 (2020)
20. Hoogeveen, D., Bennett, A., Li, Y., Verspoor, K.M., Baldwin, T.: Detecting misflagged duplicate questions in community question-answering archives. In: Twelfth international AAAI conference on web and social media (2018)
21. Imtiaz, Z., Umer, M., Ahmad, M., Ullah, S., Choi, G.S., Mehmood, A.: Duplicate questions pair detection using siamese MaLSTM. IEEE Access **8**, 21932–21942 (2020)
22. Jaccard, P.: The distribution of the flora in the alpine zone. 1. New phytologist **11**(2), 37–50 (1912)
23. Jin, J., Li, Y., Zhong, X., Zhai, L.: Why users contribute knowledge to online communities: An empirical study of an online social Q&A community.

Information & management **52**(7), 840–849 (2015)
24. Kamath, A., Gupta, S., Carvalho, V.: 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, pp. 5545–5550 (2019)
25. Kamienski, A., Bezemer, C.P.: An empirical study of Q&A websites for game developers. Empirical Software Engineering (2021)
26. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International conference on machine learning, pp. 1188–1196 (2014)
27. Li, Z., Yin, G., Yu, Y., Wang, T., Wang, H.: Detecting duplicate pull-requests in GitHub. In: Proceedings of the 9th Asia-Pacific Symposium on Internetware, pp. 1–6 (2017)
28. Li, Z., Yu, Y., Zhou, M., Wang, T., Yin, G., Lan, L., Wang, H.: Redundancy, context, and preference: An empirical study of duplicate pull requests in OSS projects. IEEE Transactions on Software Engineering (2020)
29. Liang, D., Zhang, F., Zhang, W., Zhang, Q., Fu, J., Peng, M., Gui, T., Huang, X.: 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, pp. 95–104 (2019)
30. Lopresti, D.P.: Models and algorithms for duplicate document detection. In: Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR'99 (Cat. No. PR00318), pp. 297–300. IEEE (1999)
31. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
32. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp. 3111–3119 (2013)
33. Mizobuchi, Y., Takayama, K.: Two improvements to detect duplicates in Stack Overflow. In: 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 563–564. IEEE (2017)
34. Nasehi, S.M., Sillito, J., Maurer, F., Burns, C.: What makes a good code example?: A study of programming Q&A in StackOverflow. In: 2012 28th IEEE International Conference on Software Maintenance (ICSM), pp. 25–34. IEEE (2012)
35. Niwattanakul, S., Singthongchai, J., Naenudorn, E., Wanapu, S.: Using of Jaccard coefficient for keywords similarity. In: Proceedings of the international multiconference of engineers and computer scientists, 6, pp. 380–384 (2013)
36. Omondiagbe, O.P., Licorish, S.A., MacDonell, S.G.: Features that predict the acceptability of Java and JavaScript answers on Stack Overflow. In: Proceedings of the Evaluation and Assessment on Software Engineering,

pp. 101–110 (2019)
37. Overflow, S.: About Stack Overflow. https://stackoverflow.com/company (2021). Accessed: July 25, 2021
38. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
39. Poerner, N., Schütze, H.: 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), pp. 1630–1641 (2019)
40. Porter, M.F., et al.: An algorithm for suffix stripping. Program **14**(3), 130–137 (1980)
41. Prabowo, D.A., Herwanto, G.B.: Duplicate question detection in question answer website using convolutional neural network. In: 2019 5th International Conference on Science and Technology (ICST), vol. 1, pp. 1–6. IEEE (2019)
42. Procaci, T.B., Nunes, B.P., Nurmikko-Fuller, T., Siqueira, S.W.: Finding topical experts in question & answer communities. In: 2016 IEEE 16th International Conference on Advanced Learning Technologies (ICALT), pp. 407–411. IEEE (2016)
43. Procaci, T.B., Siqueira, S.W., Nunes, B.P., Nurmikko-Fuller, T.: Modelling experts behaviour in Q&A communities to predict worthy discussions. In: 2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT), pp. 291–295. IEEE (2017)
44. Rahman, M.M., Roy, C.K.: An insight into the unresolved questions at Stack Overflow. In: 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, pp. 426–429. IEEE (2015)
45. Rakha, M.S., Bezemer, C.P., Hassan, A.E.: Revisiting the performance evaluation of automated approaches for the retrieval of duplicate issue reports. IEEE Transactions on Software Engineering **44**(12), 1245–1268 (2017)
46. Rakha, M.S., Bezemer, C.P., Hassan, A.E.: 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 **23**(5), 2597–2621 (2018)
47. Ramos, J., et al.: Using tf-idf to determine word relevance in document queries. In: Proceedings of the first instructional conference on machine learning, vol. 242, pp. 29–48. Citeseer (2003)
48. Reimers, N., Gurevych, I.: 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 (2019). URL https://arxiv.org/abs/1908.10084

49. Richardson, L.: Beautiful soup. https://www.crummy.com/software/BeautifulSoup (2020). Accessed: September 5, 2021
50. Rochette, A., Yaghoobzadeh, Y., Hazen, T.J.: Unsupervised domain adaptation of contextual embeddings for low-resource duplicate question detection. arXiv preprint arXiv:1911.02645 (2019)
51. Rodrigues, J., Saedi, C., Maraev, V., Silva, J., Branco, A.: Ways of asking and replying in duplicate question detection. In: Proceedings of the 6th joint conference on lexical and computational semantics (SEM), pp. 262–270 (2017)
52. Rücklé, A., Moosavi, N.S., Gurevych, I.: Neural duplicate question detection without labeled training data. arXiv preprint arXiv:1911.05594 (2019)
53. Runeson, P., Alexandersson, M., Nyholm, O.: Detection of duplicate defect reports using natural language processing. In: 29th International Conference on Software Engineering (ICSE'07), pp. 499–510. IEEE (2007)
54. Saedi, C., Rodrigues, J., Silva, J., Branco, A., Maraev, V.: Learning profiles in duplicate question detection. In: 2017 IEEE international conference on information reuse and integration (IRI), pp. 544–550. IEEE (2017)
55. Santos, T., Burghardt, K., Lerman, K., Helic, D.: 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, pp. 969–973 (2020)
56. Shah, D.J., Lei, T., Moschitti, A., Romeo, S., Nakov, P.: Adversarial domain adaptation for duplicate question detection. arXiv preprint arXiv:1809.02255 (2018)
57. Shen, X., Jia, A.L., Shen, S., Dou, Y.: Helping the ineloquent farmers: Finding experts for questions with limited text in agricultural Q&A communities. IEEE Access **8**, 62238–62247 (2020)
58. Silva, R.F., Paixão, K., de Almeida Maia, M.: Duplicate question detection in Stack Overflow: A reproducibility study. In: 2018 IEEE 25th international conference on software analysis, evolution and reengineering (SANER), pp. 572–581. IEEE (2018)
59. Song, K., Tan, X., Qin, T., Lu, J., Liu, T.Y.: MPNet: Masked and permuted pre-training for language understanding. arXiv preprint arXiv:2004.09297 (2020)
60. Sun, C., Lo, D., Khoo, S.C., Jiang, J.: Towards more accurate retrieval of duplicate bug reports. In: 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), pp. 253–262. IEEE (2011)
61. Sun, C., Lo, D., Wang, X., Jiang, J., Khoo, S.C.: A discriminative model approach for accurate duplicate bug report retrieval. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1, pp. 45–54 (2010)
62. Tabassum, J., Maddela, M., Xu, W., Ritter, A.: Code and named entity recognition in StackOverflow. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL) (2020). URL https://www.aclweb.org/anthology/2020.acl-main.443/

63. Viggiato, M., Lin, D., Hindle, A., Bezemer, C.P.: What causes wrong sentiment classifications of game reviews. IEEE Transactions on Games (2021)
64. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**, 261–272 (2020). DOI 10.1038/s41592-019-0686-2
65. Wang, L., Zhang, L., Jiang, J.: Detecting duplicate questions in Stack Overflow via deep learning approaches. In: 2019 26th Asia-Pacific Software Engineering Conference (APSEC), pp. 506–513. IEEE (2019)
66. Wang, L., Zhang, L., Jiang, J.: Duplicate question detection with deep learning in Stack Overflow. IEEE Access **8**, 25964–25975 (2020)
67. Wang, Q., Xu, B., Xia, X., Wang, T., Li, S.: Duplicate pull request detection: When time matters. In: Proceedings of the 11th Asia-Pacific Symposium on Internetware, pp. 1–10 (2019)
68. Wang, Y.: 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)
69. Witkowski, W.: 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` (2020). Accessed: July 4, 2021
70. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45. Association for Computational Linguistics, Online (2020). URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`
71. Wu, Y., Wang, S., Bezemer, C.P., Inoue, K.: How do developers utilize source code from Stack Overflow? Empirical Software Engineering **24**(2), 637–673 (2019)
72. Xu, B., Hoang, T., Sharma, A., Yang, C., Xia, X., Lo, D.: Post2vec: Learning distributed representations of Stack Overflow posts. IEEE Transactions on Software Engineering (2021)
73. Xu, Z., Yuan, H.: Forum duplicate question detection by domain adaptive semantic matching. IEEE Access **8**, 56029–56038 (2020)
74. Yang, X.L., Lo, D., Xia, X., Wan, Z.Y., Sun, J.L.: What security questions do developers ask? A large-scale study of Stack Overflow posts. Journal of Computer Science and Technology **31**(5), 910–924 (2016)

75. Ying, A.T.T.: Mining challenge 2015: Comparing and combining different information sources on the Stack Overflow data set. In: The 12th Working Conference on Mining Software Repositories (2015)
76. Zhang, W.E., Sheng, Q.Z., Lau, J.H., Abebe, E.: Detecting duplicate posts in programming QA communities via latent semantics and association rules. In: Proceedings of the 26th International Conference on World Wide Web, pp. 1221–1229 (2017)
77. Zhang, W.E., Sheng, Q.Z., Lau, J.H., Abebe, E., Ruan, W.: Duplicate detection in programming question answering communities. ACM Transactions on Internet Technology (TOIT) **18**(3), 1–21 (2018)
78. Zhang, W.E., Sheng, Q.Z., Shu, Y., Nguyen, V.K.: Feature analysis for duplicate detection in programming QA communities. In: International Conference on Advanced Data Mining and Applications, pp. 623–638. Springer (2017)
79. Zhang, W.E., Sheng, Q.Z., Tang, Z., Ruan, W.: Related or duplicate: Distinguishing similar CQA questions via convolutional neural networks. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 1153–1156 (2018)
80. Zhang, X., Liu, S., Chen, X., et al.: Social capital, motivations, and knowledge sharing intention in health Q&A communities. Management Decision (2017)
81. Zhang, Y., Lo, D., Xia, X., Sun, J.L.: Multi-factor duplicate question detection in Stack Overflow. Journal of Computer Science and Technology **30**(5), 981–997 (2015)
82. Zhang, Y., Lu, T., Phang, C.W., Zhang, C.: 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 **20**(8), 3 (2019)
83. Zhou, Q., Liu, X., Wang, Q.: Interpretable duplicate question detection models based on attention mechanism. Information Sciences **543**, 259–272 (2021)
84. Řehůřek, R.: Gensim: Topic modelling for humans. `https://radimrehurek.com/gensim` (2021). Accessed: September 5, 2021