A Transformer-based Approach for Augmenting Software Engineering Chatbots Datasets

Ahmad Abdellatif
University of Calgary
Calgary, Canada
ahmad.abdellatif@ucalgary.ca

Khaled Badran
Concordia University
Montreal, Canada
k_badran@encs.concordia.ca

Diego Elias Costa
Concordia University
Montreal, Canada
diego.costa@concordia.ca

Emad Shihab
Concordia University
Montreal, Canada
emad.shihab@concordia.ca

ABSTRACT

Background: The adoption of chatbots into software development tasks has become increasingly popular among practitioners, driven by the advantages of cost reduction and acceleration of the software development process. Chatbots understand users' queries through the Natural Language Understanding component (NLU). To yield reasonable performance, NLUs have to be trained with extensive, high-quality datasets, that express a multitude of ways users may interact with chatbots. However, previous studies show that creating a high-quality training dataset for software engineering chatbots is expensive in terms of both resources and time. Aims: Therefore, in this paper, we present an automated transformer-based approach to augment software engineering chatbot datasets. Method: Our approach combines traditional natural language processing techniques with the BART transformer to augment a dataset by generating queries through synonym replacement and paraphrasing. We evaluate the impact of using the augmentation approach on the Rasa NLU's performance using three software engineering datasets. Results: Overall, the augmentation approach shows promising results in improving the Rasa's performance, augmenting queries with varying sentence structures while preserving their original semantics. Furthermore, it increases Rasa's confidence in its intent classification for the correctly classified intents. Conclusions: We believe that our study helps practitioners improve the performance of their chatbots and guides future research to propose augmentation techniques for SE chatbots.

ACM Reference Format:
Ahmad Abdellatif, Khaled Badran, Diego Elias Costa, and Emad Shihab. 2024. A Transformer-based Approach for Augmenting Software Engineering Chatbots Datasets. In . ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Chatbots have proven themselves to be a game changer in a variety of domains from personal assistant to customer services. With their benefits in saving time and cost, chatbots have made significant advances in various fields [40]. The increased popularity and proven benefits of chatbots are driving software engineering (SE) practitioners to develop chatbots to help developers in various SE tasks. For example, Lin et al. [30] developed the MSABot, a chatbot that assists developers in building and managing microservices projects (e.g., setting microservices project parameters). Abdellatif et al. [2] developed the MSRBot to answer questions related to software projects (e.g., "Who fixed bug 5?").

Through natural language, chatbots enable users to communicate with different services intuitively. To understand users' queries (i.e., messages), chatbots leverage a Natural Language Understanding (NLU) component [1, 2, 30]. In essence, NLUs use AI and natural language processing techniques to extract structured information (the intent of the user's query and related entities) from unstructured input text. To use NLUs effectively, chatbot developers need to obtain or craft high-quality datasets containing a variety of user queries to train the NLU in extracting the intention behind the user's questions. Prior work shows that the performance of the NLU is directly related to the quality and diversity of the dataset used in its training [1]. Indeed, including syntactically diverse queries with the same semantics in their training datasets to train the NLU in the different ways users may ask for the same information. [13, 41]. For example, the queries "List the developers who resolved issue 5", "Who fixed bug 5?", and "Which developer fixed issue 5?" have the same semantics (identify the developer who fixed a specific bug) but different sentence structures.

Crafting a diverse and high-quality dataset is one of the most costly and time-consuming tasks in chatbot development [2, 3, 14]. Chatbot developers need to brainstorm a variety of training queries in order to familiarize the NLU with new terms (synonyms replacement) and diverse sentence structures (paraphrasing) [2, 33]. Previous studies show that the lack of high-quality datasets is a limiting factor for the efficiency of chatbots [2, 14]. For example, Dominic et al. [14] reported that the absence of training queries limited their chatbot performance. Likewise, Abdellatif et al. [2] stated that the MSRBot failed to classify some user queries correctly because of the scarcity of training data. Consequently, this data problem hinders the practitioners' ability to develop more efficient

SE chatbots, as the training dataset would need to be crafted and augmented manually. Moreover, there is a number of posts on Stack Overflow where chatbot developers ask for more data to enhance the NLUs' performance [23, 38].

When practitioners create the initial set of training queries, *Augmentation* techniques are often used to create and incorporate new training queries into the dataset [3]. This process is key for many machine learning applications where data scarcity is a major limiting factor for model's performance. A number of studies have focused on evaluating different augmentation approaches to improve different machine learning applications in the field of sentiment analysis [22, 32] and hate-speech detection [36].

Inspired by recent breakthroughs in language models for different SE tasks [11, 12, 42], we explore a transformer-based augmentation approach for SE chatbots that emulates the way in which chatbot developers augment their datasets [2, 33]. More specifically, the augmentation approach takes as input a few training queries and uses them to augment more queries by replacing some words with their synonyms (synonyms replacement). Then, it uses a fine-tuned BART transformer to change the query structure (paraphrasing). To evaluate the performance of the augmentation approach, we perform an empirical study by applying the Augmentation Approach to three well-crafted datasets that represent distinct SE-related tasks, namely 1) Repository: questions exploring software project data, 2) Ask Ubuntu: technical questions from the Ubuntu Q&A community on Stack Exchange, and 3) Stack Overflow: questions commonly asked by developers on Q&A websites. These datasets include a total of 767 queries covering 19 different intents (e.g., LookingForCodeSample). To put the Augmentation Approach results into perspective, we compare the Rasa NLU's performance without augmenting any query to the training dataset (Baseline), and augmenting human queries to the training dataset (Human). Our study is formalized through the following research questions:
RQ1: Can the augmentation approach improve the NLU's performance? Overall, the Augmentation Approach shows promising results in improving the Rasa's performance, where it marginally increases the Rasa's performance by up to 3.2% compared to Baseline. Furthermore, the augmented queries have different sentence structures with the same semantics as the input queries. In cases where there was no improvement, the Approach augments queries with small modifications, which could result in overfitting the Rasa. On the other hand, the Human augmented queries improve Rasa's performance across all datasets compared to Augmentation Approach.
RQ2: Does the augmentation approach increase the NLU's confidence in its classification? Training Rasa using the approach augmented queries increases the Rasa's confidence in its intent classification for the correctly classified intents. In some cases, the Augmentation Approach outperforms Humans in terms of confidence scores for correctly classified intents. For the misclassified intents, the results show that the augmented queries in the Human and Augmentation Approach experiments increase Rasa's confidence in the misclassified intents.

Our study makes the following contributions:

- We propose an approach that uses synonym replacement and paraphrasing techniques to augment queries for chatbots in the software engineering domain.
- We explore the impact of using the augmentation approach on the NLU's performance using three datasets from the SE domain and different use cases that vary in the number of initial training queries.
- We provide a replication package containing the implementation of the augmentation approach as a prototype tool and our results [4] to facilitate the replication and accelerate future research in the area.

2 BACKGROUND

Chatbots are software bots that interact with users through natural language [27]. This simple method of interaction is what gives chatbots their appeal and makes them a suitable conduit between users and services, such as in customer service [25]. To facilitate this chat-like interaction, modern chatbots leverage the natural language understanding (NLU) component, which extracts structured information from unstructured text (user's query). Typically, the NLU component extracts two key pieces of information from the user's query; the intent and entities. The **intent** represents the intention/goal behind the user's query, while **entities** are important keywords in the query. For example, when a user asks "*What are the fixing commits for bug 5391?*", the intent is to know which commits fixed a specific bug in the project ('FixingCommit' intent), whereas the bug number ('5391') is the entity. When interacting with a chatbot, users are free to express the same intent in different ways. For example, the queries "Show the fixing commits for issue 5391" and "What changes solved 5391?" have the same intent ('FixingCommit') but different syntax.

It is critical for chatbots to have a robust NLU that extracts the intent from the users' queries correctly as it gives the chatbot an accurate assessment of the users' intentions, leading it to take the right course of action (send a reply or perform a task). In contrast, a poorly performing NLU that misclassifies the users' intent will lead the chatbot to reply incorrectly and/or perform a wrong action which has a direct and negative impact on the satisfaction of the chatbot users [26].

When the NLU extracts an intent, it also returns a confidence score corresponding to that intent. The confidence score shows how confident the NLU is in its intent classification, and it has a value that ranges between 0 (i.e., not confident) to 1 (i.e., fully confident). Chatbot developers use the confidence score to determine whether the chatbot has understood the user's query well enough (high confidence score), in which case, the chatbot should perform an action. Otherwise, if the user's query is not clear enough (low confidence score), the chatbot asks the user to clarify the query in order for the chatbot to better understand the intent [1].

Chatbot developers brainstorm to come up with training queries at the early stages of the chatbot development cycle to train the NLU on different ways a user could ask about specific intent [2]. Then, they augment more training queries by replacing some words with their synonyms and re-writing (paraphrasing) the original training queries in different ways [3]. However, augmenting the chatbot training queries is a resource and time consuming task [2, 14].

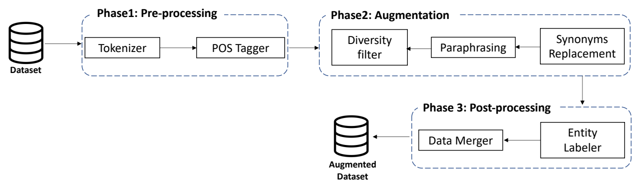


Figure 1: An overview of the augmentation approach.

In our study, we want to investigate the impact of an augmentation approach on the NLU's performance in terms of intents classification and confidence score. Intent classification with high confidence is critical to ensure that chatbots correctly understand and answer the user's question, which improves the user experience with the chatbot.

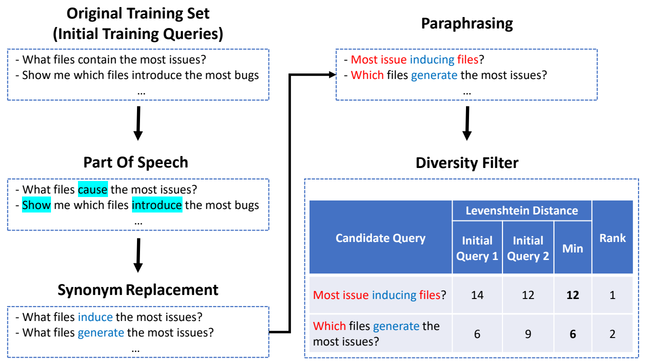
3 APPROACH

The key idea of the augmentation process is that by using a small initial set of training queries (called the original training set) as input, we can generate new queries that retain the same semantics while having different terms/keywords and brand new sentence structures. Figure 1 shows the components of the augmentation approach. Initially, the augmentation approach takes a query from the original training set as an input, then it tokenizes the query and extracts the part-of-speech for each token in the query. Next, the augmentation approach generates new queries (candidate queries) by introducing new keywords and paraphrasing the input query. Then, it filters the candidate queries and keep only the queries with the highest potential of improving the NLU's performance. Finally, the approach labels the entities in the selected queries and merges them with the original training set to obtain the final (augmented) training set. In this section, we detail each component of the augmentation approach. Also, we showcase an end-to-end example (Figure 2) to demonstrate how each component works.

Tokenizer: Commonly used in an NLP pipeline, we start our augmentation approach by tokenizing the input data, to better process and augment text, such as identifying part of speech and replacing synonyms. Thus, in this component, we split each input query in the original training set into tokens using a pre-trained model from the SpaCy library.

Part-of-Speech (POS) Tagger: This component identifies the POS (e.g., verb, noun, adjective) for each token in the query. This makes the augmentation approach more flexible as it can apply synonyms replacement on specific POS tokens. For example, in case the dataset already has diverse synonyms for noun tokens, then the chatbot developers may want to diversify the dataset by having more synonyms of verbs. The working example in Figure 2 showcases the POS tagging component identifying and labelling the verb tokens (i.e., cause, show, and introduce) in the input queries in the original training set.

Synonyms Replacement: Given the tagged tokens from the POS Tagger component, the Synonyms Replacement component replaces certain tokens (e.g., verbs, nouns) with their synonyms to obtain new candidate queries. The goal here is to familiarize the NLU with a large variety of similar terms that might appear in the users' queries, but have not been exposed to the NLU during


Original Training Set
(Initial Training Queries)

- What files contain the most issues?
- Show me which files introduce the most bugs

...

Part Of Speech

- What files cause the most issues?
- Show me which files introduce the most bugs

...

Synonym Replacement

- What files induce the most issues?
- What files generate the most issues?

...

Paraphrasing

- Most issue inducing files?
- Which files generate the most issues?

...

Diversity Filter

Candidate Query	Levenshtein Distance			Rank
	Initial Query 1	Initial Query 2	Min	
Most issue inducing files?	14	12	**12**	1
Which files generate the most issues?	6	9	**6**	2

Figure 2: A working example of the augmentation approach.

its training. To obtain the list of synonyms for a token, one could use any of the available thesauruses such as WordNet [19] and PyDictionary [8]. However, those are general purposes thesauruses and not tailored for SE specific terminologies. For example, when looking for synonyms to the term 'bug', the WordNet thesauruses returns 'germ', 'microbe', and 'hemipteron'. Since our goal is to augment the SE chatbot training dataset, we opted to use a thesaurus that is specialized for SE to capture the specific language and terminologies used in the SE domain. Therefore, the Synonyms Replacement component leverages an SE thesaurus, which is a word2Vec model trained on Stack Overflow posts to capture the SE terms [17]. The SE thesaurus returns 'issue' and 'error' as synonyms to the term 'bug'.

This component creates a new candidate query by replacing one token within a query from the original training set with its synonym. In case a query has two or more tokens to be replaced, the Synonyms Replacement component generates new candidate queries based on all the possible combinations of the replaceable tokens' synonyms. In the working example in Figure 2, the Synonyms Replacement component replaces the verb token 'cause' from the query "What files cause the most issues?" with its synonyms 'induce' and 'generate', thus creating two new candidate queries (e.g., "What files induce the most issues?").

In our preliminary analysis, we find that replacing nouns with their synonyms generates too much noise. For example, the synonym of the 'developer' and 'button' tokens are replaced with 'prismic' and 'buttom'; respectively. Thus, we opt to just replace verb tokens with their synonyms in our study.

Paraphrasing: One aspect of expanding the training dataset is to expose the NLU to new terminologies. The other important aspect is training NLU on a variety of sentence structures for queries. Chatbot developers typically paraphrase the queries they add to the training set because users can phrase the same question in different ways [3]. In fact, this process is recommended by the NLU vendors to enhance their performance in intent classification [13, 41].

Therefore, the Paraphrasing component diversifies the sentence structure of candidate queries while preserving their meaning (intent).

The paraphrasing component takes as input each candidate queries from the Synonyms Replacement component. To paraphrase queries, this component leverages the recent transformer based neural machine translation (Seq2Seq) model called BART [29]. BART is a general language model proposed by Facebook AI and has been used by prior work for paraphrasing task [16, 44, 48] as it achieves state-of-art performance in various NLP tasks (e.g., machine translation, summarization, and text generation) [29]. BART is trained through corrupting the input example (e.g., delete one of its tokens) during the training stage and then predicting the correct form of the corrupted sentence. We fine-tune BART to perform paraphrasing task (discussed in Section 4).

In the working example (Figure 2), the Paraphrasing component takes the two candidate queries from the Synonyms Replacement component as an input and outputs paraphrased queries (e.g., "Most issue inducing files?"). The final output of the Paraphrasing component is a list of new candidate queries that preserve the intent and have both new terms and different sentence structures compared to the original training set.

Diversity Filter: The main goal of the augmentation approach is to generate candidate queries with the highest potential of improving the NLU's performance. Therefore, the Diversity Filter selects the candidate queries yielded by the Paraphrasing component that are most syntactically different compared to the original training set. This helps to mitigate the issue of overfitting the NLU that may occur when the candidate queries do not increase the syntactical diversity of the original training set.

As a means to measure the diversity, this component computes the Levenshtein distance [28] (number of edits between two queries) between the candidate queries and the original training set. The higher the Levenshtein distance, the more dissimilar the queries. In the working example (Figure 2), the Levenshtein distance between the candidate query "Most issue inducing files?" and the original training query "What files contain the most issues?" is 14.

Then, the Diversity Filter ranks the candidate queries based on their minimum Levenshtein distance to any query in the original training set. In other words, the Diversity Filter ranks candidate queries by the highest minimum Levenshtein distance from the original query, placing queries that are more syntactically different from the original query at the top. This approach balances the inclusion of queries that are syntactically diverse but still convey the same semantic intent as the original query. Finally, the top N candidate queries are kept by the Diversity Filter while the rest are discarded (N is configurable). In other words, the Diversity Filter component discards candidate queries that are syntactically similar to those that are already present in the original training set. For instance, if we set N=1 in our showcase example, the candidate query "Most issue inducing files?" passes the filter because it has the highest minimum Levenshtein distance (12).

Entity Labeler: Typically, the chatbot training datasets include annotations of both intents and entities for all queries in the set. Such annotations are essential for some NLUs for the intents classification step [1]. However, the candidate queries that are retained after the Diversity Filter do not contain any entity annotations. Hence, the Entity Labeler component uses heuristics to label the entities in the candidate queries. To establish the heuristics, we examined 400 random samples from different intents generated by the Paraphrasing component and found that the entities remain the same or experience minor modifications only during the paraphrasing. For example, the FileName entity (e.g., 'ConsumerRecords') could be changed (e.g., 'Consumer Records') during the paraphrasing. The only exception here is the DateTime entities, where a specific date (e.g., 21-12-2022) can be changed to 'last week'. Based on our observations, we define heuristics to label entities in the candidate queries. Therefore, the Entity Labeler component reads all labeled entities in the original training dataset and automatically labels the entities in the candidate queries using the defined heuristics.

Data Merger: The output of any augmentation approach should be a training dataset that is ready for use. The Data Merger component is responsible for adding the output queries of the augmentation approach to their corresponding intents in the original training set to generate augmented training set file. Therefore, chatbot practitioners can use the output file to train the NLU used in their chatbots.

4 EVALUATION SETUP

In this section, we present the setup of our case study to evaluate the impact of the proposed augmentation approach on the NLU's performance. We detail our selection of the SE datasets used in the evaluation, NLU platform used for training and testing, tuning the BART transformer, and experiment design.

4.1 Datasets

To evaluate how effective the approach is in augmenting a variety of SE datasets, we select three distinct datasets: Repository [2], Ask Ubuntu [10], and Stack Overflow [46]. Our dataset selection is based on three criteria. First, these datasets represent realistic questions that software practitioners ask about software projects and development. Second, they have adequate numbers of training and testing queries (ten or more queries per intent) to conduct proper evaluation. Table 1 presents the intents in each of the datasets, their definitions, and the distribution of queries in the training and test sets corresponding to each intent. Finally, the datasets are publicly available and have been used in previous studies [1, 24, 39]. In the following, we provide a description of each dataset.

Repository: Contains questions asked by software practitioners about their software repositories to the MSRBot [2]. One example query from this dataset is "What are the commits that introduce bug HHH8492?". For this dataset, the MSRBot developers created the training set manually and composed the test set from queries asked by the MSRBot users. Thus, the training and test sets in this corpus originate from a real-life use case of an SE-based chatbot in practice. The Repository dataset contains 398 queries belonging to ten intents in total.
Ask Ubuntu: This dataset was constructed using the most popular posts from the Ubuntu Q&A community on Stack Exchange, one of the most popular online discussion forums [10]. Braun et al. [10] selected the most popular questions, which were then annotated using Amazon Mechanical Turk. An example of a query from this dataset is "How to upgrade Ubuntu 14.04.1 to 14.04.2". This dataset contains 154 queries split into four intents. It is important to note that we discarded the 'Other' intent because it had an insufficient number of queries (i.e., three queries) for our evaluation.

Table 1: Performance comparison results for augmentation approach against the baseline and human.

Dataset	Intent	Definition	Train	Test	Total
Repository	**BuggyCommitsByDate**	Present the buggy commit(s) which happened during a specific time period.	66	13	79
	BuggyCommit	Identify the bugs that are introduced because of certain commits.	52	9	61
	BuggyFiles	Determine the most buggy files in the repository to refactor them.	37	13	50
	FixCommit	Identify the commit(s) which fix a specific bug.	31	11	42
	BuggyFixCommits	Identify the fixing commits that introduce bugs at a particular time	32	7	39
	CountCommitsByDates	Identify the number of commits that were pushed during a specific time period.	11	21	32
	ExperiencedDevFixBugs	Identify the developer(s) who have experience in fixing bugs related to a specific file.	15	14	29
	OverloadedDev	Determine the overloaded developer(s) with the highest number of unresolved bugs.	15	9	24
	FileCommits	View details about the changes that occurred on a file.	10	12	22
	CommitsByDate	Present the commit information (e.g., commit message) at a specific time.	8	12	20
Ask Ubuntu	**SoftwareRecommendation**	Looking for applications that perform specific task (e.g., extract images from PDF).	17	40	57
	MakeUpdate	Looking for information related to upgrading Ubuntu version to a newer version.	10	37	47
	ShutdownComputer	Related fix shutdown issues in Ubuntu OS.	13	14	27
	SetupPrinter	Setup a printer and fix printer installation issues.	10	13	23
Stack Overflow	**LookingForCodeSample**	Looking for information related to implementation (e.g., code snippets).	66	66	132
	UsingMethodImproperly	An improper use of a method is causing unexpected behaviour.	25	26	51
	LookingForBestPractice	Looking for the recommended (best) practice, approach or solution for a problem.	6	6	12
	FacingError	Facing an error or a failure in a program, mostly in the form of an error message.	5	5	10
	PassingData	Passing data between different frameworks or method calls.	5	5	10

Stack Overflow: Contains labeled software development questions from Stack Overflow [46], another popular development Q&A website. The queries in this dataset were collected by Ye et al. [46] and then Abdellatif et al. [1] labeled the queries' intents. In total, this dataset is composed of 215 queries and five different intents. One example query from this dataset is "How can I get Font X offset width in java2D?".

4.2 NLU

The goal of the augmentation approach is to improve the NLU's performance by augmenting a given training dataset. Hence, we need to select an NLU platform to train its model and perform our evaluation. For this study, we select Rasa, an open-source NLU platform developed by Rasa Technologies [35]. Unlike third-party NLUs that operate on the cloud (e.g., Google Dialogflow), Rasa can be installed, configured, and run locally, which consumes fewer resources. And the Rasa implementation stays the same during our experiment, while the internal implementation of third-party NLUs might change without any prior notice to the users [1]. Moreover, Rasa has been used by prior work to develop SE chatbots [2, 15, 30]. In our implementation, we used Rasa version 2.5 as it was the latest stable version available at the time of our study.

4.3 BART Tuning

As discussed in Section 3, the paraphrasing component uses BART transformer to paraphrase the candidate queries resulted from the Synonyms Replacement component. BART is trained on 160GB of documents (e.g. Wikipedia, news articles, stories) with a sentence reconstruction loss [29]. To use BART, we need to fine-tune it to the specific task at hand which is, in our case, to paraphrase text. Therefore, we use the following three datasets to fine-tune BART: 1) Quora Question Pairs contains over 149,263 lines of potential duplicate pairs of questions obtained from Quora social Q&A website, 2) Microsoft Research Paraphrase composed of 3,749 pairs of sentences extracted from the internet (e.g., news sources) and then annotated by humans to indicate whether each pair captures the same semantics, and 3) Paraphrase Adversaries from Word Scrambling contains 25,368 pairs of paraphrased sentences generated using word swapping and back-translation created by Zhang et al. [47]. These three datasets have been used in prior work for text paraphrasing [16, 44], which makes them a solid choice to fine-tune BART. Furthermore, we use the *BART-large* model, which has 12 layers in the encoder and decoder, and more than 374 million of parameters. We train the BART model on a cloud with 6 core Intel E5-2683 v4 Broadwell, 64GB of RAM, and NVIDIA V100 Volta GPU

Table 2: Performance comparison results for augmentation approach against the baseline and human.

Dataset	No. of Queries/Intent	Baseline F1-score	Augmentation Approach F1-score	Augmentation Approach % Improvement	Augmentation Approach % Optimal	Human F1-score	Human % Improvement
Repository	One	43.7	44.7*	2.3	6.9	58.1*	33.0
	Three	62.8	64.3*	2.4	34.1	67.2*	7.0
	Five	66.4	68.5*	3.2	60.0	69.9*	5.3
Ask Ubuntu	One	71.6	73.5*	2.7	17.3	82.6*	15.4
	Three	84.1	83.6*	-0.6	-	90.4*	7.5
	Five	87.1	84.2*	-3.3	-	90.2*	3.6
Stack Overflow	One	32	32.2	0.6	2.5	40*	25
	Three	36.9	37.9*	2.7	12.3	45*	22.0

* The difference is statistically significant (p-value<0.05) compared to Baseline.

(32G HBM2 memory). We examined BART's output with different numbers of returned paraphrases (i.e., 3, 5, 7, and 10) and found that BART performs best in terms of having diverse sentence structure and preserving the semantics of the input when it returns three paraphrases at most for any input query.

4.4 Evaluation Settings

To evaluate the impact of using the augmentation approach on the NLU's performance, we train the NLU after augmenting the original training set and then evaluate the NLU's performance using the test set. Unlike the Repository and Ask Ubuntu datasets, there is no predefined train-test split in the Stack Overflow dataset. Therefore, we divide the Stack Overflow dataset into 50%-50% for training and test splits through random stratified sampling. Table 1 presents the distribution of queries for each intent in the training and test splits in the Repository, Ask Ubuntu, and Stack Overflow datasets. It is important to note that we repeat this step 10 times to obtain different inputs (training queries) and evaluate the performance of the augmentation approach on diverse input queries.

After obtaining the training and test splits for all datasets, we craft three scenarios: A scenario with 1 query/intent, 3 queries/intent, and 5 queries/intent. The goal of our scenarios is to evaluate the potential of the augmentation approach in scenarios with little training dataset where augmentation have the tendency to yield the biggest impact on the NLU's performance. For each scenario, we randomly select the queries per intent from the original training dataset. For example, the Stack Overflow dataset has five intents. Thus, the 3 queries/intent scenario in the Stack Overflow dataset has in total of 15 training queries (i.e., 3 queries per intent).

4.5 Performance Evaluation

To assess the NLU's performance in classifying intents, we compute the widely used metrics of precision, recall, and F1-score. Precision is the percentage of the correctly classified queries to the total number of classified queries for that intent (i.e., Precision = $\frac{TP}{TP+FP}$). The recall is calculated as the percentage of the correctly classified queries to the total number of queries for that intent in the test set (i.e., Recall = $\frac{TP}{TP+FN}$). To have an overall score, we combine both the precision and recall using the F1-score weighted by class' support (weighted F1-score), which has been used in similar studies [1, 7, 21]. More specifically, we start by computing the F1-score (i.e., F1-score = $2 \times \frac{Precision \times Recall}{Precision + Recall}$) for all classes. Then, we aggregate all F1-scores using the weighted average, with the class' support as weights. It is important to note that, although we evaluate all three metrics, we only present the weighted F1-score in the paper. We make the precision, recall, and F1-score results for each intent publicly available online [4].

5 RESULTS

In this section, we present the results of our case study with respect to the three research questions. For each research question, we present the motivation for the question, detail the approach to answer the question, and present the results.

5.1 RQ1: Can the augmentation approach improve the NLU's performance?

Motivation: Previous studies show that augmenting the training set used to train the NLU leads to improving its performance [1]. However, the process of crafting and collecting more training data is done manually, which is a costly and time-consuming task [2, 15]. The goal of this research question is to evaluate the impact of using the augmentation approach (discussed in Section 3) on the NLU's performance. Succeeding in this task saves time and resources, which allows practitioners to focus on the critical tasks of their chatbots rather than augmenting the training set.
Approach: To answer this RQ and put the results of the augmentation approach into perspective, we perform three different experiments:
(1) Baseline: Establishes a baseline for the NLU's performance. In this experiment, we use the original queries from each of the scenarios (i.e., 1, 3, and 5 queries/intent) to train the NLU. Hence, no augmented query is included in the NLU training.
(2) Augmentation Approach: This experiment reflects the situation where a chatbot practitioner augments the original training dataset using our augmentation approach. In this experiment, we apply the augmentation approach to scenarios 1, 3, and 5 queries/intent. Then, we augment the queries resulting from the augmentation approach into the scenario set. In our study, we set the configuration to augment one query per intent N=1 such that the Diversity Filter keeps only the top-ranking candidate query. For example, augmenting a scenario of 1 query/intent using the augmentation approach yields a training set of 2 queries/intent. This makes our evaluation manageable since it requires manually examining all the generated queries for all intents across the three different datasets to obtain

Table 3: Sample of the augmented queries by the augmentation approach.

Original Query	Augmented Queries	Intent
Show me the number of commits happened last week	How many commits did you commit between last week and this week?	CommitsCountByDate
Show me the classes which introduced the most of bug	What files are the ones that added most of bugs	BuggyFiles
How can one shutdown a PC using the keyboard?	Hotkey to shut down from login screen?	ShutdownComputer
Python inserting variable string as file name	How do I insert a string as a file name in Python?	LookingForCodeSample
Spring 4.0.2 schema error	What is the reason behind the schema error in Spring 4.0.2?	FacingError

insights about the augmented queries. Also, it makes our study manageable in terms of consumption power since our approach requires running two deep learning models, BART transformer and NLU, to generate queries.

(3) Human: In this experiment, we evaluate the impact of using queries crafted by humans to augment the original training set on the NLU's performance. This experiment reflects a situation where a chatbot practitioner starts with a set of training queries and then manually augments the training set. The human-augmented queries are high-quality and can most likely improve the NLU's performance compared to any augmentation approach. In this experiment, from the dataset discussed in Section 4.1, we randomly select one query per intent from the training split that was not used in the scenarios (e.g., 3 queries per intent) and augment the selected query to the scenario. Thus, we have the same settings as the Augmentation Approach experiment (i.e., augmenting one query per intent).

We follow the same evaluation steps for all datasets, experiments, and scenarios. First, we train the NLU using the scenario's queries resulting from the experiment. Next, we use the test set to evaluate the NLU's performance for each experiment and record the results. It is important to note that in the Stack Overflow dataset, we only run our evaluation on scenarios 1 query/intent and 3 queries/intent. This is because some intents (e.g., 'PassingData') in the 5 queries/intent scenario of the Stack Overflow dataset do not have enough training queries left to be used as input queries to randomly select in the Human experiment.

To measure the improvement in the NLU's performance achieved by a specific experiment (i.e., Augmentation Approach and Human) to the baseline, we resort to the %Improvement metric. The %Improvement for an experiment (EXP) is measured using the equation:

$$\%Improvement_{(Exp)} = \frac{F1\,Score_{(Exp)} - F1\,Score_{(Baseline)}}{F1\,Score_{(Baseline)}} * 100.$$

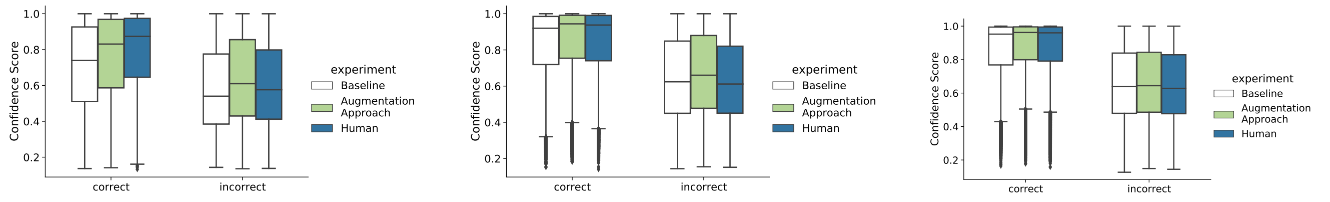
Since the main idea of the augmentation approach is to imitate the chatbot practitioner on how they augment their training datasets, we use %Optimal metric to measure how close the augmentation approach is to the human augmentation performance. This measure calculates the ratio of the %Improvement achieved by the Augmentation approach to the %Improvement achieved by the Human experiment as follows:

$$\%Optimal = \frac{\%Improvement_{(Augmentation\,Approach)}}{\%Improvement_{(Human)}} * 100$$

It is worth noting that we repeat the evaluation step 100 times for each scenario and experiment to reduce the randomness effect of the NLU's model. We report the average results from those runs in our results. We use the non-parametric Mann-Whitney U test [45] to statistically compare the difference between the results of all experiments. We chose the Mann-Whitney U test because it does not assume any specific distribution of the data.

Results: Table 2 presents the F1-scores, %Improvement, and %Optimal for Baseline, Augmentation Approach, and Human experiments on all scenarios and datasets. From the table, we observe that the augmentation approach marginally improves the NLU's performance (%Improvement > 2%) in six scenarios. On the other hand, there are two scenarios where the augmentation approach does not improve the performance of the NLU (%Improvement ≤ 0). For example, applying the augmentation approach on the scenario of 1 query/intent in the Ask Ubuntu dataset decreases the NLU's performance (%Improvement < -0.06). Among the six scenarios where there is an improvement in the NLU's performance, we notice that the augmentation approach is most effective in the scenarios that have more queries. For example, the augmentation approach achieves similar results to Human in 5 queries/intent scenario for the Repository dataset with %Optimal of 60%. This is because those scenarios (e.g., 5 queries/intent) have more queries that are used to augment more diverse queries compared to the scenarios with fewer queries (e.g., 1 query/intent).

Upon closer examination of the augmented queries across all scenarios and datasets, we have two main observations: 1) Some of the augmented queries have different sentence structures than the queries in the original training set. Table 3 presents a sample of the original training queries and their corresponding queries augmented through the augmentation approach. For instance, the augmentation approach augments the query "Show me the number of commits happened last week" from the original training set to a new candidate query "How many commits did you commit between last week and this week?". In some cases, the augmented queries make minor changes to the original queries. For example, the augmentation approach replaces the word 'fix' with 'remedy' in the initial training query "Who has the most bugs to fix?" to generate a new candidate query "Who has the most bugs to remedy?". 2) Upon manual examination of all augmented queries across different scenarios and datasets, we find that the augmentation approach preserves the semantic (intent) of the original queries as shown in Table 3. We further examined all generated queries (4,380 queries) by Paraphrasing component to investigate whether BART transformer preserves the intent of the original queries before passing the queries to the Diversity Filter component. We find that 99.4% of paraphrased queries maintain the intent of their original queries, with only 0.6% (26 out of 4,380) of queries having a changed intent across all runs in different scenarios and datasets. For example,



(a) 1 query/intent (b) 3 queries/intent (c) 5 queries/intent

Figure 3: The confidence score distributions for scenarios 1, 3, and 5 queries/intent in the Repository dataset.

the initial training query "Give me the changes of ConnectRestConfigurable file" asks about presenting the changes that touch ConnectRestConfigurable file (FileCommits intent), however, the augmented query is "How can I change the changes of ConnectRestConfigurable file", which changes the meaning of the query to ask about modifying the commits that changed the file.

To better understand the factors that the augmentation approach yields to low performance for scenarios with 3 and 5 queries/intent in the Ask Ubuntu dataset, we examined the augmented queries of all runs in those scenarios. Surprisingly, we find that the augmentation approach often augments queries with small modifications to the initial training query. For example, the approach modified the Ubuntu version in "How to upgrade Ubuntu [14.04.1](UbuntuVersion) to [14.04.2](UbuntuVersion)?" to augment a new query "How to upgrade from Ubuntu [10.10](UbuntuVersion) to [11.04](UbuntuVersion)?". In other cases, the approach omits some text from the initial training queries. For example, in the "How To Install [Canon LBP2900B](Printer) printer in [14.04 LTS](UbuntuVersion)? I tried the method for LBP2900 but it didnt work" query, the approach removed "I tried the method for LBP2900 but it didnt work" to create the augmented query (i.e., "How To Install [Canon LBP2900B](Printer) printer in [14.04 LTS](UbuntuVersion)? I tried the method for LBP2900 but it didnt work"). Thus, the high similarity between the augmented queries and the original training queries might cause overfitting and decrease the performance of the NLU. On the other hand, we observe that the performance for 3 and 5 queries/intent in the Ask Ubuntu dataset is high (F1-score > 84%), indicating that the augmentation approach is particularly beneficial in situations where the performance of NLU is low, which is the main goal of our study (i.e., improve the low performance of the NLU).

Our study results show that the augmentation approach has a small, yet positive impact on the performance of the NLU. We reiterate that this improvement is achieved through automation and does not require human input. Additionally, we believe that the generated queries by the approach could serve as inspiration for chatbot practitioners to diversify sentence structures and synonyms while manually expanding the training dataset.

Augmented queries provide small but significant improvement in the NLU's performance. In various scenarios, the approach is most efficient when augmenting training datasets of 3 and 5 queries/intent. Additionally, it augments queries with varying sentence structures while preserving their original semantics.

5.2 RQ2: Does the augmentation approach increase the NLU's confidence in its classification?

Motivation: Each intent classification performed by the NLU has a corresponding confidence score, as discussed in Section 2. Chatbot developers rely on the confidence score returned with the classified intent to determine the chatbot's next action [1]. More specifically, developers tend to design their chatbots so that, if the NLU is not confident in its intent classification, the chatbot asks for further clarification from the user (i.e., "Sorry, I did not understand your question, could you please rephrase it?"). Developers expect a well-trained NLU to provide higher confidence scores for correctly classified intents and lower scores for misclassified intents to minimize wrong actions performed by the chatbot. One way to increase the NLU's confidence in its classification is to train the NLU on more queries for each intent (i.e., augment the training set). Therefore, in this RQ, we evaluate the impact of using the augmentation approach on the NLU's confidence, particularly with regard to the confidence scores returned with the correctly and incorrectly classified intents.

Approach: To answer this RQ, we use the output of the three experiments (Baseline, Augmentation Approach, and Human) described in RQ1. In particular, we examine the confidence scores returned by the NLU for the test set queries across all the experiments. By using the three experiments, we establish a baseline (Baseline experiment) for how confident the NLU is in its intent classification. Then, we use that baseline to measure the impact of using the augmentation approach on the NLU's confidence (Augmentation experiment) compared to the impact of using human-augmented queries (Human experiment). We hypothesize that augmenting more queries increases the NLU's confidence scores for the correctly classified intents while decreasing its confidence for the misclassified intents. This is because the NLU will be exposed to more ways (i.e., different syntactic structures) of asking about the same intent. Finally, to compare the confidence scores for the correctly and incorrectly classified intents, we present the distributions of confidence scores for each case. Also, we perform the non-parametric unpaired Mann-Whitney U test [45] to verify whether the difference in the confidence scores across the experiments' results (e.g., Baseline vs Augmentation Approach experiments) are statistically significant.

Results: Figure 3 shows the confidence score distributions for both the correctly and incorrectly classified intents for all scenarios in the Repository dataset. As shown in the figure, in all experiments, we find that the median confidence scores of correctly classified intents are higher than the misclassified intents. In fact, these results are in-line with the ones in the prior work [1].

From the figure, we also observe that using the Augmentation Approach significantly increases the NLU's confidence in its intent classification for correctly classified intents, particularly in the 1 query/intent scenario, compared to the Baseline across all scenarios. For example, in 1 query/intent scenario, the Augmentation approach increases the median confidence scores (76.3%) by 6% compared to the Baseline (70.1%). In some cases, the Augmentation Approach outperforms the Human experiment in terms of confidence scores for correctly classified intents. For example, in the 3 queries/intent scenario, the Baseline experiment has a median confidence score of 83.1%, while the Augmentation approach and Human experiment have median confidence scores of 85% and 84.3%, respectively.

For incorrectly classified intents, we find that the augmentation approach increases the NLU's confidence on misclassified queries compared to the Baseline. That is, the median confidence scores for incorrectly classified intents is higher when training the NLU with the augmented dataset. Surprisingly, using human-augmented queries (i.e., Human experiment) increases the overall confidence scores for misclassified intents compared to the baseline. For example, in the 1 query/intent scenario, the median confidence score is 58.1% for the Baseline, while the median confidence scores for the Augmentation approach and Human experiments are 63.3% and 60.3%, respectively.

Using the Augmentation Approach increases the NLU's confidence in intent classification for correctly classified intents across scenarios. Nonetheless, the augmented queries in the Human and Augmentation Approach experiments increase the NLU's confidence in misclassified intents.

6 DISCUSSION

Quality of generated data is far more important than quantity. NLU achieves reasonable performance even when trained on a few queries per intent, as discussed in RQ1. Results show that with only one training query per intent (i.e., 1 query/intent scenario), the NLU reaches an F1-Score ranging from 32% in the Stack Overflow and up to 71.6% in the Ask Ubuntu datasets (see Table 2) in the Baseline experiment. This observation is further reinforced when looking at the baseline performance with 5 queries/intent scenario, where the NLU achieves an F1-score up to 87% in the Ask Ubuntu dataset. Although this seems promising for chatbot developers, achieving a robust NLU performance for real-scenario is still very challenging, as chatbots may deal with sensitive data (e.g., software project data) or performs critical tasks (e.g., merging vulnerable code into the main branch). Given this level of robustness of modern NLUs, augmentation approaches should focus on the quality of the generated queries more than just its ability to increase the size of the training dataset. Furthermore, having high-quality augmented queries inspires chatbot practitioners to add a variety of new queries to their dataset, ultimately resulting in improved performance.

Practitioners need to consider fine-tuning the confidence threshold of their chatbots. The confidence score determines the chatbot's response, with high scores resulting in action (e.g., answering the user's question) and low scores prompting the chatbot to request clarification from the user in order to improve its understanding of the user's intent [1]. The Augmentation approaches raise the confidence levels of NLUs, both for correctly and incorrectly classified intents, as discussed in RQ2. Therefore, we recommend practitioners who use augmentation approaches consider fine-tuning the appropriate confidence score threshold of their chatbots to accept the classified intent or trigger the fallback action. Moreover, our results suggest that future augmentation approaches should take into account the confidence scores for both correctly and incorrectly classified intents when augmenting queries.

There is a need for a well-crafted dataset of paraphrased queries for the SE domain. In our study, we configured BART to return three paraphrases, as discussed in Section 4. In RQ1 results, we observe that BART tends to generate queries with only slight variations between them. For example, the "How to move content from QListWidget to a QStringList with PyQt4?" query in the Stack Overflow dataset, BART generates "How to moving content from QListWidget to a QStringList with PyQt4?", "How to moves content from QListWidget to a QStringList with PyQt4?", and "How do I move content from QListWidget to a QStringList with PyQt4?", which have high similarities between the queries. We reiterate that in fine-tuning the BART model to paraphrase queries, we reuse three datasets (e.g., Quora question pairs) that have been used in prior work [16, 44]. None of these datasets are related to the SE domain. The lack of a more specialized SE dataset might limit BART's ability to generate paraphrased queries specific to SE. To the best of our knowledge, there is no crafted dataset that contains pairs of paraphrased queries related to the SE domain. We plan (and encourage others) to create benchmarks that contain paraphrased pairs of queries representing different SE tasks in order to fully utilize the potential transformers in the SE context.

that only one intent had new examples added to it. However, we also observed that the performance of other intents dropped at the same time. And while this can potentially be due to the randomness of the model, it can also indicate that adding examples to one intent can affect the performance of other intents.

7 RELATED WORK

The goal of this paper is to propose an approach to augment the training datasets for the SE chatbots. Thus, we divide the prior work into two main areas; work related to developing SE chatbots and work related to dataset augmentation.

Software Engineering Chatbots. A plethora of studies propose chatbots to assist developers in their daily tasks [2, 9, 15, 34]. For example, Bradley et al. [9] propose Devy, a chatbot to help software developers in their development tasks (e.g., pushing the new code changes to the project repository). Dominic et al. [15] develop a chatbot to help newcomers with the onboarding process on OSS projects. Abdellatif et al. [2] develop a chatbot to answer questions related to software projects. Paikari et al. [34] propose a chatbot, called Sayme, that resolves code conflicts among the software teams. Lin et al. [30] propose MSA chatbot to assist practitioners in developing and maintaining the micro-services architecture.

The increased usage of software chatbots in the software engineering domain motivates our work; to improve SE chatbot's performance and help practitioners focus on the chatbot core functionalities rather than brainstorming more training queries. However, our work differs in that we propose an augmentation approach and do not develop a new chatbot.

Datasets Augmentation. There is a number of studies that examine data augmentation techniques for text classification [5, 20, 22, 32, 36, 43]. For example, Marivate & Sefara [32] evaluate the impact of four augmentation approaches (WordNet-based synonym, Word2vec-based, Round Trip Translation, and Mixup augmentation) on the performance of the classification algorithm using Sentiment 140, AG News, and Hate Speech datasets. Sharifirad et al. [37] propose an approach to improve the classification of sexiest tweets. In particular, the authors generate new tweets by replacing the words with their synonyms using the ConceptNet and Wikidata. The results show that applying the proposed approach improves the machine learning models in text classifications. Feng et al. [20] evaluate different augmentation approaches (e.g., random insertion) to fine-tune GPT-2 for text generation task using Yelp reviews dataset. The results show that keyword replacement and character-level synthetic noise are effective for text augmentation. Imran et. al [22] evaluate the impact of data augmentation techniques (e.g., word insertion) to improve emotion classification. The work closest to ours is the work that augments datasets to enhance NLU's performance [31]. Malandrakis et al. [31] investigate the use of neural generative encoder-decoder models to improve the NLU's performance trained on movie and Live entertainment datasets.

To the best of our knowledge, there is no work that studied evaluating the data augmentation approach that combines synonym replacement and paraphrasing techniques, and tailored it to improve the NLU's performance for SE chatbots. Our work differs and complements the prior work in two ways. First, our work imitates the chatbot developers by adding new synonyms and changing the sentence structure of the original training set. Second, we evaluate the approach using three SE datasets. Overall, our work complements the studies augmenting the training dataset to enhance NLU's performance by providing an augmentation approach for the SE domain.

8 THREATS TO VALIDITY

In this section, we discuss the threats to internal, replicability, and external validity of our study.

Internal Validity. Concerns confounding factors that could have influenced our results. We configure BART to return three queries (sequences), which might impact the quality of the paraphrased queries. To alleviate this threat, the first two authors examined the output from BART using different numbers of returned queries (1, 3, 5, 7, and 10) and found that configuring the returned queries to be three yields the best results in terms sentence structure diversity and preserving the semantics. Another threat to internal validity relates to the manual labelling of our datasets, which could introduce subjectivity bias. However, these datasets have been used in many prior works in SE [1, 18]. Another potential threat is the choice of steps for our proposed approach and evaluation settings, which could impact the results. For example, using metrics other than Levenshtein distance (e.g., [6]) to measure the diversity of the candidate queries might impact the results. Therefore, we plan to evaluate using a wider range of settings (e.g., more scenarios) in the future. In our study, we used three datasets (e.g., Quora Question Pairs) to fine-tune the BART transformer, as discussed in Section 4.3. Fine-tuning a transformer model involves some degree of randomness (e.g., random initialization of weights). Thus, replicating the study might lead to different results. We mitigate this issue by providing the scripts and datasets used in the evaluation in our replication package [4]. We believe that our study serves as a starting point for chatbot practitioners to leverage transformers in augmenting chatbot training dataset.

External Validity Concerns the generalization of our findings. In this study, we evaluate an augmentation approach using three different datasets from the SE domain. The results might not generalize to other SE datasets. However, these datasets have been used by prior work to evaluate the performance of different NLUs and propose chatbots in the SE domain [1, 15, 30]. Another threat is that we use Rasa NLU for evaluation; hence our results might not be generalizable to other NLUs. However, we select Rasa as it is an open-source NLU which guarantees that its internal implementation stays the same during our entire study. Moreover, Rasa has been widely used by practitioners to develop SE chatbots [2, 15, 30].

9 CONCLUSION & FUTURE WORK

Software chatbots play important roles in the software engineering domain, enabling practitioners to perform various software development tasks, such as running tests, through natural language. The NLU component is essential for chatbots to understand the user's input. Training the NLU on possible queries from users is critical because it impacts user satisfaction with the chatbot. However, prior work shows that creating and augmenting a high-quality training dataset for SE chatbots is a costly and time-consuming task. To help chatbot developers improve the NLU's performance of their chatbots, we evaluate a transformer-based augmentation approach that emulates the standard way practitioners augment their chatbot training set. More specifically, the augmentation approach augments more queries by replacing words with synonyms (synonyms replacement) and paraphrasing the query using BART transformer. We evaluate the impact of using the augmentation approach on the NLU's performance using three datasets that represent distinct SE-related tasks. We find that the augmented queries provide a small but significant improvement in the NLU's performance. Moreover, the augmentation approach augments queries with diverse sentence structures while maintaining their original semantics (intents). Also, our results show that the augmentation approach increases the NLU's confidence in both correctly and incorrectly classified intents.

Our paper outlines directions for future work in this area. First, we plan to examine the performance of different transformers (e.g., GPT2, BERT, RoBERTa) in the task of paraphrasing SE queries. Also, we intend to investigate the use of Stack Overflow posts to craft a dataset containing paraphrased queries to help tune the transformers for the SE domain. Finally, we plan to evaluate the augmentation approach using more SE datasets and NLU platforms.

REFERENCES

[1] Ahmad Abdellatif, Khaled Badran, Diego Costa, and Emad Shihab. 2021. A Comparison of Natural Language Understanding Platforms for Chatbots in Software Engineering. *IEEE Transactions on Software Engineering (TSE)* (2021), 1–1. https://doi.org/10.1109/TSE.2021.3078384

[2] Ahmad Abdellatif, Khaled Badran, and Emad Shihab. 2020. MSRBot: Using Bots to Answer Questions from Software Repositories. *Empirical Software Engineering (EMSE)* 25 (2020), 1834–1863. Issue 3.

[3] Ahmad Abdellatif, Diego Elias Costa, Khaled Badran, Rabe Abdelkareem, and Emad Shihab. 2020. Challenges in Chatbot Development: A Study of Stack Overflow Posts. In *Proceedings of the 17th International Conference on Mining Software Repositories (MSR'20)*. To Appear.

[4] Diego Costa Emad Shihab Ahmad Abdellatif, Khaled Badran. 2024. A Transformer-based Approach for Augmenting Software Engineering Chatbots Datasets. https://zenodo.org/records/11121853. (Accessed on 05/06/2024).

[5] Ali Amin-Nejad, Julia Ive, and Sumithra Velupillai. 2020. Exploring Transformer Text Generation for Medical Dataset Augmentation. In *Proceedings of the 12th Language Resources and Evaluation Conference*. European Language Resources Association, Marseille, France, 4699–4708.

[6] Kevin Bache, David Newman, and Padhraic Smyth. 2013. Text-based measures of document diversity. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Chicago, Illinois, USA) *(KDD '13)*. Association for Computing Machinery, New York, NY, USA, 23–31. https://doi.org/10.1145/2487575.2487672

[7] Guy Barash, Eitan Farchi, Ilan Jayaraman, Orna Raz, Rachel Tzoref-Brill, and Marcel Zalmanovici. 2019. Bridging the Gap between ML Solutions and Their Business Requirements Using Feature Interactions. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Tallinn, Estonia) *(ESEC/FSE 2019)*. Association for Computing Machinery, New York, NY, USA, 1048–1058. https://doi.org/10.1145/3338906.3340442

[8] Pradipta Bora. [n. d.]. geekpradd/PyDictionary: PyDictionary is a Dictionary Module for Python 2/3 to get meanings, translations, synonyms and antonyms of words. https://github.com/geekpradd/PyDictionary. (Accessed on 07/01/2024).

[9] Nick C. Bradley, Thomas Fritz, and Reid Holmes. 2018. Context-Aware Conversational Developer Assistants. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. Association for Computing Machinery, New York, NY, USA, 993–1003. https://doi.org/10.1145/3180155.3180238

[10] Daniel Braun, Adrian Hernandez Mendez, Florian Matthes, and Manfred Langen. 2017. Evaluating Natural Language Understanding Services for Conversational Question Answering Systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*. Association for Computational Linguistics, Saarbrücken, Germany, 174–185. https://doi.org/10.18653/v1/W17-5522

[11] Matteo Ciniselli, Nathan Cooper, Luca Pascarella, Antonio Mastropaolo, Emad Aghajani, Denys Poshyvanyk, Massimiliano Di Penta, and Gabriele Bavota. 2022. An Empirical Study on the Usage of Transformer Models for Code Completion. *IEEE Transactions on Software Engineering* 48, 12 (2022), 4818–4837. https://doi.org/10.1109/TSE.2021.3128234

[12] Matteo Ciniselli, Luca Pascarella, and Gabriele Bavota. 2022. To What Extent Do Deep Learning-Based Code Recommenders Generate Predictions by Cloning Code from the Training Set?. In *Proceedings of the 19th International Conference on Mining Software Repositories* (Pittsburgh, Pennsylvania) *(MSR '22)*. Association for Computing Machinery, New York, NY, USA, 167–178. https://doi.org/10.1145/3524842.3528440

[13] Microsoft Docs. 2022. Good example utterances. https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-concept-utterance. (Accessed on 03/30/2024).

[14] James Dominic, Jada Houser, Igor Steinmacher, Charles Ritter, and Paige Rodeghero. 2020. Conversational Bot for Newcomers Onboarding to Open Source Projects. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (Seoul, Republic of Korea) *(ICSEW'20)*. Association for Computing Machinery, New York, NY, USA, 46–50. https://doi.org/10.1145/3387940.3391534

[15] James Dominic, Jada Houser, Igor Steinmacher, Charles Ritter, and Paige Rodeghero. 2020. Conversational Bot for Newcomers Onboarding to Open Source Projects. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (Seoul, Republic of Korea) *(ICSEW'20)*. Association for Computing Machinery, New York, NY, USA, 46–50. https://doi.org/10.1145/3387940.3391534

[16] Thomas Dopierre, C. Gravier, and Wilfried Logerais. 2021. ProtAugment: Unsupervised diverse short-texts paraphrasing for intent detection meta-learning. In *The 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, Vol. abs/2105.12995.

[17] Vasiliki Efstathiou, Christos Chatzilenas, and Diomidis Spinellis. 2018. Word Embeddings for the Software Engineering Domain. In *Proceedings of the 15th International Conference on Mining Software Repositories* (Gothenburg, Sweden) *(MSR '18)*. Association for Computing Machinery, New York, NY, USA, 38–41. https://doi.org/10.1145/3196398.3196448

[18] Farbod Farhour, Ahmad Abdellatif, Essam Mansour, and Emad Shihab. 2024. A Weak Supervision-Based Approach to Improve Chatbots for Code Repositories. In *Proceedings of the ACM International Conference on the Foundations of Software Engineering (FSE'24)*.

[19] Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books. https://mitpress.mit.edu/9780262561167/

[20] Steven Y. Feng, Varun Gangal, Dongyeop Kang, Teruko Mitamura, and Eduard Hovy. 2020. GenAug: Data Augmentation for Finetuning Text Generators. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*. Association for Computational Linguistics, Online, 29–42. https://doi.org/10.18653/v1/2020.deelio-1.4

[21] A. Ilmania, Abdurrahman, S. Cahyawijaya, and A. Purwarianti. 2018. Aspect Detection and Sentiment Classification Using Deep Neural Network for Indonesian Aspect-Based Sentiment Analysis. In *2018 International Conference on Asian Language Processing (IALP)*. 62–67. https://doi.org/10.1109/IALP.2018.8629181

[22] Mia Mohammad Imran, Yashasvi Jain, Preetha Chatterjee, and Kostadin Damevski. 2023. Data Augmentation for Improving Emotion Recognition in Software Engineering Communication. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (Rochester, MI, USA) *(ASE '22)*. Association for Computing Machinery, New York, NY, USA, Article 29, 13 pages. https://doi.org/10.1145/3551349.3556925

[23] jsphdnl. 2023. nlp Conversational Data for building a chat bot. https://stackoverflow.com/questions/45821517/conversational-data-for-building-a-chat-bot. (Accessed on 01/19/2024).

[24] Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction. 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)*. Association for Computational Linguistics, Hong Kong, China, 1311–1316. https://doi.org/10.18653/v1/D19-1131

[25] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky. 2018. Software Bots. *IEEE Software* 35, 1 (2018), 18–23. https://doi.org/10.1109/MS.2017.4541027

[26] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky. 2018. Software Bots. *IEEE Software* 35, 1 (2018), 18–23. https://doi.org/10.1109/MS.2017.4541027

[27] Carlene Lebeuf, Alexey Zagalsky, Matthieu Foucault, and Margaret-Anne Storey. 2019. Defining and Classifying Software Bots: A Faceted Taxonomy. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, 1–6. https://doi.org/10.1109/BotSE.2019.00008

[28] Vladimir I Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* 10 (February 1966), 707.

[29] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

[30] Chun-Ting Lin, Shang-Pin Ma, and Yu-Wen Huang. 2020. MSABot: A Chatbot Framework for Assisting in the Development and Operation of Microservice-Based Systems. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (Seoul, Republic of Korea) *(BotSE'20)*. Association for Computing Machinery, New York, NY, USA, 36–40. https://doi.org/10.1145/3387940.3391501

[31] Nikolaos Malandrakis, Minmin Shen, Anuj Goyal, Shuyang Gao, Abhishek Sethi, and Angeliki Metallinou. 2019. Controlled Text Generation for Data Augmentation in Intelligent Artificial Agents. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*. Association for Computational Linguistics, Hong Kong, 90–98. https://doi.org/10.18653/v1/D19-5609

[32] Vukosi Marivate and Tshephisho Sefara. 2020. Improving Short Text Classification Through Global Augmentation Methods. In *Machine Learning and Knowledge Extraction*, Andreas Holzinger, Peter Kieseberg, A Min Tjoa, and Edgar Weippl (Eds.). Springer International Publishing, Cham, 385–399.

[33] Microsoft. 2023. Utterances - Azure Cognitive Services | Microsoft Docs. https://docs.microsoft.com/en-us/azure/cognitive-services/luis/concepts/utterances. (Accessed on 01/08/2024).

[34] Elahe Paikari, JaeEun Choi, SeonKyu Kim, Sooyoung Baek, MyeongSoo Kim, SeungEon Lee, ChaeYeon Han, YoungJae Kim, KaHye Ahn, Chan Cheong, and André van der hoek. 2019. A Chatbot for Conflict Detection and Resolution. In *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*. 29–33. https://doi.org/10.1109/BotSE.2019.00016

[35] Rasa. [n. d.]. Conversational AI Platform | Superior Customer Experiences Start Here. https://rasa.com/. (Accessed on 07/01/2024).

[36] Georgios Rizos, Konstantin Hemker, and Björn Schuller. 2019. Augment to Prevent: Short-Text Data Augmentation in Deep Learning for Hate-Speech Classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 991–1000. https://doi.org/10.1145/3357384.3358040

[37] Sima Sharifirad, Borna Jafarpour, and Stan Matwin. 2018. Boosting Text Classification Performance on Sexist Tweets by Text Augmentation and Text Generation Using a Combination of Knowledge Graphs. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Association for Computational Linguistics, Brussels, Belgium, 107–114. https://doi.org/10.18653/v1/W18-5114

[38] Sheri. 2022. Python Intent classification for Chatbot. https://stackoverflow.com/questions/62970861/intent-classification-for-chatbot. (Accessed on 04/28/2024).

[39] Kumar Shridhar, Harshil Jain, Akshat Agarwal, and Denis Kleyko. 2020. End to End Binarized Neural Networks for Text Classification. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*. Association for Computational Linguistics, Online, 29–34. https://doi.org/10.18653/v1/2020.sustainlp-1.4

[40] Margaret-Anne Storey and Alexey Zagalsky. 2016. Disrupting Developer Productivity One Bot at a Time. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Seattle, WA, USA) *(FSE 2016)*. Association for Computing Machinery, New York, NY, USA, 928–931. https://doi.org/10.1145/2950290.2983989

[41] Tmbo. 2022. multiple entity recognition· Issue #427 · RasaHQ/rasa. https://github.com/RasaHQ/rasa/issues/427. (Accessed on 03/23/2024).

[42] Rosalia Tufano, Simone Masiero, Antonio Mastropaolo, Luca Pascarella, Denys Poshyvanyk, and Gabriele Bavota. 2022. Using Pre-Trained Models to Boost Code Review Automation. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 2291–2302. https://doi.org/10.1145/3510003.3510621

[43] Jason Wei and Kai Zou. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. 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)*. Association for Computational Linguistics, Hong Kong, China, 6382–6388. https://doi.org/10.18653/v1/D19-1670

[44] Peter West, Ximing Lu, Ari Holtzman, Chandra Bhagavatula, Jena D. Hwang, and Yejin Choi. 2021. Reflective Decoding: Beyond Unidirectional Generation with Off-the-Shelf Language Models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 1435–1450. https://doi.org/10.18653/v1/2021.acl-long.114

[45] Daniel S Wilks. 2011. *Statistical methods in the atmospheric sciences*. Vol. 100. Academic press.

[46] Deheng Ye, Zhenchang Xing, Chee Yong Foo, Zi Qun Ang, Jing Li, and Nachiket Kapre. 2016. Software-Specific Named Entity Recognition in Software Engineering Social Content. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 1. 90–101. https://doi.org/10.1109/SANER.2016.10

[47] Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase Adversaries from Word Scrambling. In *Proc. of NAACL*.

[48] Jianing Zhou, Hongyu Gong, and Suma Bhat. 2020. PIE: A Parallel Idiomatic Expression Corpus for Idiomatic Sentence Generation and Paraphrasing. In *The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021), MWE Workshop, 2021.*