VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI-590018

Seminar Report

On

"A Comparison of Natural Language Understanding Platforms for Chat-bots in Software Engineering"

Submitted in partial fulfillment for the award of degree

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

APOORVA A
4KV19CS009

Under the supervision of

Prof. Bhavya P S

Assistant Professor

Dept. of CS&E

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K.V.G. COLLEGE OF ENGINEERING SULLIA, D.K. - 574 327
2022-23

K.V.G. COLLEGE OF ENGINEERING

(AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI)

SULLIA, D.K. 574 327

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the seminar project work entitled ***"A Comparison of Natural Language Understanding Platforms for Chat-bots in Software Engineering"*** *carried out by* ***APOORVA A*** *bearing USN* ***4KV19CS009****, the bonafide student of KVG College of Engineering in partial fulfillment for the award of degree of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the year 2022-23. It is certified that all corrections/suggestions indicated for internal assessment of Seminar has been incorporated in this report and has been approved as it satisfies the academic requirements in respect of seminar work prescribed for said degree.*

Seminar Guide
Prof. Bhavya P S
Assistant Professor
Dept. of CS&E
KVGCE, Sullia

Head of the Department
Dr. Ujwal U J
Professor &Head
Dept. of CS&E
KVGCE, Sullia

Seminar Co-ordinators **Signature**

1. **Dr. Savitha C K**
2. **Prof. Yashaswini K L**

ACKNOWLEDGEMENT

I am deeply indebted to the Architect of modern Sullia, Late **Dr. KURUNJI VENKATRAMANA GOWDA**, Founder President, Academy of Liberal Education Sullia, for having been a constant source of inspiration.

I am extremely grateful to **Dr. RENUKA PRASAD K V**, President, Governing Council for his constant support throughout the course of the seminar work.

I gratefully acknowledge for the help rendered by our beloved Principal **Dr. SURESHA V,** for providing constant encouragement and support throughout the course.

I am also thankful to **Dr. UJWAL U J**, Professor and Head, Department of Computer Science and Engineering, CEO, KVGCE Sullia for his guidance, support, cooperation and motivation.

I am also thankful to my seminar guide **Prof. Bhavya P S,** Assistant Professor**,** Department of Computer Science and Engineering for her inspiration and guidance.

I record esteemed gratitude to seminar coordinators **Dr. SAVITHA C K**, Professor, Department of Computer Science and Engineering and **Prof. YASHASWINI K L**, Assistant Professor, Department of Computer Science and Engineering for their inspiration and guidance.

I express my sincere thanks to all teaching and non-teaching staff members of our college for their valuable support and co-operation. Finally, I would like to thank my parents for their encouragement with love and affection.

ABSTRACT

Chat-bots are envisioned to dramatically change the future of Software Engineering, allowing practitioners to chat and inquire about their software projects and interact with different services using natural language. At the heart of every chat-bot is a Natural Language Understanding (NLU) component that enables the chat-bot to understand natural language input. Recently, many NLU platforms were provided to serve as an off-the-shelf NLU component for chat-bots, however, selecting the best NLU for Software Engineering chat-bots remains an open challenge.Therefore, in this paper, we evaluate four of the most commonly used NLUs, namely IBM Watson, Google Dialog-flow, Rasa, and Microsoft LUIS to shed light on which NLU should be used in Software Engineering based chat-bots. Specifically, we examine the NLUs' performance in classifying intents, confidence scores stability, and extracting entities. To evaluate the NLUs, we use two datasets that reflect two common tasks performed by Software Engineering practitioners, 1) the task of chatting with the chat-bot to ask questions about software repositories 2) the task of asking development questions on Q&A forums (e.g., Stack Overflow). According to our findings, IBM Watson is the best performing NLU when considering the three aspects (intents classification, confidence scores, and entity extraction). However, the results from each individual aspect show that, in intents classification, IBM Watson performs the best with an F1-measure>84%, but in confidence scores, Rasa comes on top with a median confidence score higher than 0.91. Our results also show that all NLUs, except for Dialog flow, generally provide trust-able confidence scores. For entity extraction, Microsoft LUIS and IBM Watson outperform other NLUs in the two SE tasks. Our results provide guidance to software engineering practitioners when deciding which NLU to use in their chat-bots.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Software chat bots are increasingly used in the Software Engineering (SE) domain since they allow users to interact with platforms using natural language, automate tedious tasks, and save time/effort . This increase in attention is clearly visible in the increase of number of bots related publications conferences , and workshops .A recent study showed that one in every four OSS projects(26%) on GitHub are using software bots for different tasks. This is supported by the fact that bots help developers perform their daily tasks more efficiently, such as deploy builds , update dependencies , and even generate fixing patches .At the core of all chat bots lie the Natural Language Understanding platforms referred hereafter simply as NLU. NLUs are essential for the catboat's ability to understand and act on the user's input. The NLU uses machine learning and natural language processing (NLP) techniques to extract structured information (the intent of the user's query and related entities) from unstructured user's input (textual information).

As developing an NLU from scratch is very difficult because it requires NLP expertise, chat bot developers resort to a handful of widely-used NLUs that they leverage in their chatbots . As a consequence of the diversity of widely-used NLUs, developers are faced with selecting the best NLU for their particular domain. This is a non-trivial task and has been discussed heavily in prior work (especially since NLUs vary in performance in different contexts) . For instance, in the context of the weather domain, Canonico and De Russis showed that IBM Watson outperformed other NLUs, while Gregori evaluated NLUs using frequently asked questions by university students and found that Dialog flow performed best. In fact, there is no shortage of discussions on Stack Over flow about the best NLU to use in chatbot implementation as choosing an unsuitable platform for a particular domain deeply impacts the user satisfaction with the chatbot .

CHAPTER 2

LITERATURE SURVEY

Storey et al. 2016[1],**“Disrupting developer productivity one bot at a time”** in this paper is to provoke and inspire researchers to study the impact (positive and negative) of Bots on software development. We outline the modern Bot landscape and use examples to describe the common roles Bots occupy in software teams. We propose a preliminary cognitive support framework that can be used to understand these roles and to reflect on the impact of Bots in software development on productivity. Finally, we consider challenges that Bots may bring and propose some directions for future research.

Xu et al. 2016[2],**“Answer Bot: Automated generation of answer summary to developers technical questions”** In this work, we aim to help developers who want to quickly capture the key points of several answer posts relevant to a technical question before they read the details of the posts. We formulate our task as a query-focused multi-answer-posts summarization task for a given technical question. Our proposed approach AnswerBot contains three main steps : 1) relevant question retrieval, 2) useful answer paragraph selection, 3) diverse answer summary generation. To evaluate our approach, we build a repository of 228,817 Java questions and their corresponding answers from Stack Overflow. We conduct user studies with 100 randomly selected Java questions (not in the question repository) to evaluate the quality of the answer summaries generated by our approach, and the effectiveness of its relevant question retrieval and answer paragraph selection components.

Abdellatif et al. 2019[3],**“MSRBot: Using bots to answer questions from software repositories”** in this paper, we use bots to automate and ease the process of extracting useful information from software repositories. While it might seem at first that applying bots on software repositories is the same as using them to answer questions based on Stack Overflow posts, the reality is that there is a big difference between the two. One fundamental difference is the fact that bots that are trained on Stack Overflow data can provide general answers, and will never be able to answer project-specific questions such as "how many bugs were opened against my project today?". Also, we would like to better understand how bots can be applied on software repository data and highlight what is and what is not achievable using bots on top of software repositories.

Author	Title	Yea r	METHODOLOGY
M.-A. Storey and A. Zagalsky.	Disrupting developer productivity one bot at a time	2016.	Outline the modern Bot landscape and use examples to describe the common roles Bots occupy in software teams. We propose a preliminary cognitive support framework that can be used to understand these roles and to reflect on the impact of Bots in software development on productivity.
B. Xu, Z. Xing, X. Xia, and D. Lo	Answer Bot: Automated generation of answer summary to developers technical questions	2016	Aim to help developers who want to quickly capture the key points of several answer posts relevant to a technical question before they read the details of the posts. We formulate our task as a query-focused multi-answer-posts summarization task for a given technical question.
A. Abdellatif, K. Badran, and E. Shihab	MSRBot: Using bots to answer questions from software repositories	2019	In this paper, we use bots to automate and ease the process of extracting useful information from software repositories. While it might seem at first that applying bots on software repositories is the same as using them to answer questions based on Stack Overflow posts, the reality is that there is a big difference between the two.

CHAPTER 3

IMPLEMENTATION

METHODOLOGY:

Explanatory Example

To demonstrate how chat-bots utilize NLUs to answer a user's query, we showcase an example of a user asking a repository related question to a chat-bot as shown in Fig. 3.1. In this example, we use a simplified architecture of the chat-bot for illustration purposes. The NLU is trained on the queries (intents) related to mining software repositories and is trained to extract repository entities from users' questions, such as a Jira Ticket (e.g., HHH-8593). In this example, after the customary greeting from the chat-bot, the user asks the chat-bot "Fix ticket 8983?" which is forwarded to the NLU where it classifiers the user's question as having a Get Maintainers intent with a confidence score of 0.06. The low confidence score (lower than a predetermined threshold) triggers the fallback intent, thus, the chat-bot asks the user to rephrase the question in a more understandable way (i.e., "Sorry, I did not understand your question, could you please rephrase the question?"). After the user rephrases the question "Which commit fixed the bug ticket 8983?", the NLU extracts the entity 'ticket 8983' of type Jira Ticket and classifies the intent of the query as Get Fixing Commits with a confidence score of 0.85. Finally, the chat-bot performs the necessary action, querying the database to answer the posed question.

Software chat-bots are the conduit between their users and automated services. Through natural language, users ask the chatbot to perform specific tasks or inquire about a piece of information. Internally, a chatbot then uses the NLU to analyze the posed query and act on the users' request. The main goal of an NLU is to extract structured data from unstructured language input. In particular, it extracts intents and entities from users' queries: intents represent the user intention/purpose of the question, while entities represent important pieces of information in the query.

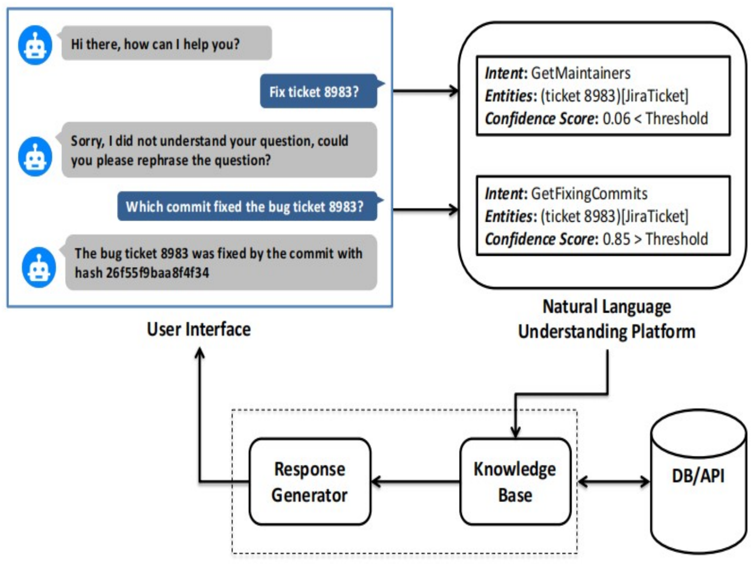

Hi there, how can I help you?
Fix ticket 8983?
Sorry, I did not understand your question, could you please rephrase the question?
Which commit fixed the bug ticket 8983?
The bug ticket 8983 was fixed by the commit with hash 26f55f9baa8f4f34
User Interface
Intent: GetMaintainers
Entities: (ticket 8983)[JiraTicket]
Confidence Score: 0.06 < Threshold
Intent: GetFixingCommits
Entities: (ticket 8983)[JiraTicket]
Confidence Score: 0.85 > Threshold
Natural Language Understanding Platform
Response Generator
Knowledge Base
DB/API

Fig. 3.1. An overview of user-chat-bot interaction.

3.1 CASE STUDY SETUP

Since the main goal of this paper is to evaluate the performance of different NLUs using SE tasks, we need to select the candidate NLUs that we want to examine and the SE tasks' data corpus to train and test those NLUs. In this section, we detail our selection of the NLUs, SE tasks used in the evaluation, and our experiment design.

3.1.1 Evaluated NLUs

There exists several widely-used NLUs that are easily integrated with third-party applications. To make our study comprehensive, we choose to examine the performance of four NLUs, namely IBM Watson, Dialog flow, Rasa, and LUIS. We select these NLUs since they are popular and widely used by both researchers and practitioners, and have been studied by prior NLU comparison work in other domains . Moreover, all selected NLUs can be trained by importing the data through their user interface or API calls, which facilitates the training process. In the following, we provide a description of those NLUs.

- **Watson Conversation (IBM Watson):** An NLU provided by IBM. IBM Watson has pre built models for different domains (e.g. banking) and a visual dialog editor to simplify building the dialog by nonprogrammers.

- **Dialog flow:** An NLU developed by Google [29]. Dialog flow supports more than 20 spoken languages and can be integrated with many chatting platforms such as Slack .

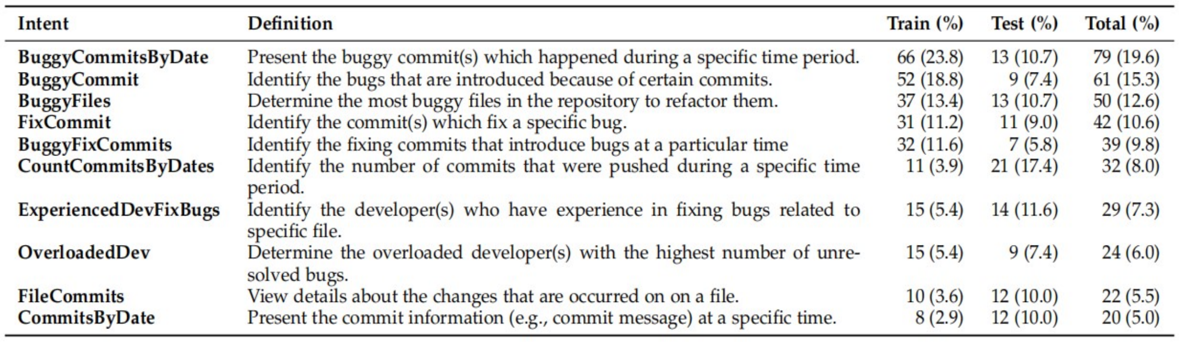
- **Rasa:** The only open-source NLU in our study, owned by Rasa Technologies . Rasa allows developers to configure, deploy, and run the NLU on local servers. Thus, increasing the processing speed by saving the network time compared to cloud-based platforms. In our evaluation, we use Rasa-nlu v0.14, which was the latest version when conducting the experiment.

- **Language Understanding Intelligent Service (LUIS):** An NLU cloud platform from Microsoft . LUIS has several pre built domains such as music and weather, and supports five programming languages: C#, Go, Java, Node.js, and Python.

3.1.2 SE Tasks and Data Corpora

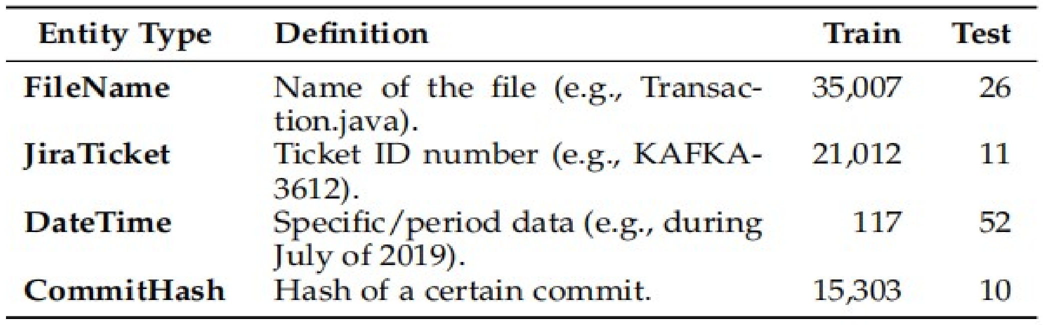
To evaluate the performance of the NLUs in the Repository and Stack Overflow tasks, we select two representative data corpora, one for each task **1) Repository corpus** used for the Repository task and includes questions posed to a chat-bot by practitioners looking for information related to their projects' software repositories **2) Stack Overflow corpus** used for the Stack Overflow task and contains a set of posts from Stack Overflow discussion threads. Our selection of these two tasks was motivated by two main reasons: Firstly, both tasks reflect realistic situations, as in both, developers are asking questions about issues they face or to get more information about their projects (e.g., fifixing commit for a bug). In fact, the Repository task also covers questions that are commonly asked by project managers to grasp the state of the project repository. Hence, both tasks make our results more generalize to the chat-bots practitioners in the SE domain. Secondly, using two tasks in our evaluation gives us better insights on how each NLU performs in different sub-contexts of SE. The Stack Overflow task uses a corpus that has a diverse set of posts from the top 8 tags in Stack Overflow, the most popular Q&A website in the developers community . On the other hand, the Repository task contains project specific information (i.e., “who touched file x?”) rather than general programming questions.

Repository Corpus. This corpus was originally used to train and test the MSRBot ,a chat-bot tailored for answering users’ questions on project repository data. This corpus contains a set of 398 queries, with 10 different intents, as shown in Table 1. Each intent contains a set of different questions with the same semantic, indicating the different ways a user could ask the same query. Those intents have questions related to code repository (e.g., “List me the changes done in Class A.java” from the File Commits intent), issue tracker (e.g., “Who has the most bug assignments?” from the OverloadedDev intent), or a combination of both code and issue tracker (e.g., “Which commits fixed HHH10956?” a query from the Fix-commit intent). The corpus includes explicitly defined training and test sets that are labeled by the developers of the MSRBot. The training set includes the questions used to train the chat-bot, acquired and curated by three developers involved in the project, while the test set is composed of questions posed by 12 software developers to test the MSRBot on a specified set of 10 different tasks. We use the training and test set as defined by the MSRBot authors in this experiment.Each entity in the Repository corpus contains a so-called list feature [40, 41, 42, 43], a list of equivalent synonyms used by the NLU to recognize entities. With the list feature, the NLUs are limited to extract the exact match of the entities, but can use the specified synonyms in the extraction process. For instance, the entity ‘HHH-7325’ of type Jira Ticket has a list of synonyms containing ‘bug7325’, ‘issue7325’, and ‘HHH7325’, hence, any of these words can be used by the NLU to recognize the entity ‘HHH-7325’. The Repository corpus contains four entity types using the list feature as shown in Table 2, covering the main artifacts in a repository, such as files (File Name), commits (Commit Hash), and tickets from the JIRA bug-tracker (Jira Ticket).



Intent	Definition	Train (%)	Test (%)	Total (%)
BuggyCommitsByDate	Present the buggy commit(s) which happened during a specific time period.	66 (23.8)	13 (10.7)	79 (19.6)
BuggyCommit	Identify the bugs that are introduced because of certain commits.	52 (18.8)	9 (7.4)	61 (15.3)
BuggyFiles	Determine the most buggy files in the repository to refactor them.	37 (13.4)	13 (10.7)	50 (12.6)
FixCommit	Identify the commit(s) which fix a specific bug.	31 (11.2)	11 (9.0)	42 (10.6)
BuggyFixCommits	Identify the fixing commits that introduce bugs at a particular time	32 (11.6)	7 (5.8)	39 (9.8)
CountCommitsByDates	Identify the number of commits that were pushed during a specific time period.	11 (3.9)	21 (17.4)	32 (8.0)
ExperiencedDevFixBugs	Identify the developer(s) who have experience in fixing bugs related to specific file.	15 (5.4)	14 (11.6)	29 (7.3)
OverloadedDev	Determine the overloaded developer(s) with the highest number of unresolved bugs.	15 (5.4)	9 (7.4)	24 (6.0)
FileCommits	View details about the changes that are occurred on on a file.	10 (3.6)	12 (10.0)	22 (5.5)
CommitsByDate	Present the commit information (e.g., commit message) at a specific time.	8 (2.9)	12 (10.0)	20 (5.0)

Table 1: Intents distribution in the Repository task.



Entity Type	Definition	Train	Test
FileName	Name of the file (e.g., Transaction.java).	35,007	26
JiraTicket	Ticket ID number (e.g., KAFKA-3612).	21,012	11
DateTime	Specific/period data (e.g., during July of 2019).	117	52
CommitHash	Hash of a certain commit.	15,303	10

Table 2: Entities distribution in the Repository task.

Stack Overflow Corpus. Stack Overflow is a popular Q&A website among developers and plays an important role in the software development life cycle . Given its importance, data from Stack Overflow has already been used in prior work to train chatbots . The titles of the questions in Stack Overflow represent a request for information by software practitioners (e.g., "How to create an JS object from scratch using a HTML button?"). Recently, Ye et al. developed a machine learning approach to extract software entities (e.g., the programming language name) from Q&A websites like Stack Overflow, and manually labeled entities from 297 Stack Overflow posts as shown in Table 3. We use the same corpus and extract the title of each post with its labeled entities. Unlike the Repository corpus, entities in the Stack Overflow corpus do not have a list feature, (list of entity synonyms), and need to be predicted by the NLUs (i.e., prediction feature). In other words, we train the NLUs only on the entities included in the training set. This allows us to evaluate the NLUs' ability to extract entities that they have not been trained on before, which emulates real-life scenarios where it is difficult for the practitioners to train the NLUs on all SE entities. While the original Stack Overflow corpus contains manually labeled entities, it lacks the intent behind the posed questions. Hence, we need to manually label the intents of the queries before we can use the corpus for the Stack Over-flow task. To achieve that, the first author used thematic analysis to categorize those queries (titles) according to their intents. Thematic analysis [45] is a technique to extract common themes within a corpus via inspection, which was done manually in our case. This method is frequently used in qualitative research to identify patterns within collected data and has been used in prior work in the SE domain . Initially, the first author categorized the queries into 19 intents. Then, we merged the categories that have a small number of examples (less than 10) but have a very similar rationale. This is because some NLUs recommend a training set of at least 10 queries for each intent [48]. For example, the queries of Facing Exception and Facing Error intents are quite similar in their goal as developers are looking for a solution to fix the crash and error. To validate the manually extracted intents, we asked two additional annotators - the second author and one other Ph.D. student - to independently label the queries using the extracted intents. For each question, we asked the annotators to evaluate whether the query has a clear intent or not using the multiple choice (‘Yes’, ‘May be’, and ‘No’). If the annotators answer the previous question with ‘Yes’ or ‘May be’, then they classify the title using one of the defined intents shown in Table 4. After both annotators finished the labeling process, we merged the labeled queries into one set to be used in the NLUs evaluation. We then use the Cohen’s Kappa coefficient to evaluate the level of agreement between the two authors . The Cohen’s Kappa coefficient is a well-known statistic that evaluates the inter-rater agreement level for categorical scales. The resulting coefficient is a scale that ranges between -1.0 and +1.0, where a negative value means poorer than chance agreement, zero indicates agreement by chance, and a positive value indicates better than chance agreement. We find that the level of agreement between the two annotators on the occurrence of intent (i.e., whether a title has an intent or not) is +0.74, and the agreement on the intents classification is +0.71. Both agreements are considered to be substantial inter-rater agreements . All three annotators discussed the disagreements and voted to the best fitting intent. After the merge, we discarded queries with unclear intent (total of 82), such as “J Console Web Application”. The final set includes 215 queries , Tables 3 and 4 show the number of entities and intents included in our evaluation, respectively. Our manual classification led to the creation of 5 intents shown in Table 4 with their definitions. The intents in the Stack Overflow Corpus represent different types of questions posted on Stack Overflow. For example, the Facing Error intent contains questions asking for a solution for an encountered error: “PHP MySQL query returns empty error message” . Table 3 shows entities used in the Stack Overflow corpus that are very specific to the SE domain. For example, Interlanguage entity type has a set of different programming languages such as Python and JavaScript. Such entities can be used by code-centrist chatbots , such as chatbots that extract files that call a particular method in the code .

Entity	Definition	Total
ProgLanguage	Types of programming languages (e.g., Java) [37].	96
Framework	Tools/frameworks that developers use (e.g., Maven) [37].	85
Standards	"Refers to data formats (e.g., JSON), design patterns (e.g.,Abstract Factory), protocols (e.g., HTTP), technology acronyms (e.g., Ajax)" [37].	20
API	An API of a library (e.g., ArrayList) [37].	67
Platform	Software/Hardware platforms (e.g., IOS) [37].	13

Table 3: List of the Stack Overflow task entities.

Intent	Description	Total (%)
LookingForCodeSample	Looking for information related to implementation. This includes looking for code snippets, the functionality of a method, or information specific to the user's needs.	132 (61.3%)
UsingMethodImproperly	A method or a framework is being used improperly causing an unexpected or unwanted behaviour of the program in hand. This can be related to code bugs or to performance issues.	51 (23.7%)
LookingForBestPractice	Looking for the recommended (best) practice, approach or solution for a problem.	12 (5.6%)
FacingError	Facing an error or a failure in a program, mostly in the form of an error message or a build failure.	10 (4.7%)
PassingData	Passing data between different frameworks or method calls.	10 (4.7%)

Table 4: List of the Stack Overflow task intents

3.3 Performance Evaluation of NLUs

We use the corpora from the Repository and Stack Overflow tasks to train IBM Watson, Dialog flow, Rasa, and LUIS and evaluate their performance. As each task has specific characteristics, we detail in the following how we train and test the NLUs for each task. To evaluate the NLUs on the Repository task, we use the same training set from the Repository corpus, which includes 10 intents with their queries and entities with their lists of synonyms. To set up the NLUs, we configure the NLUs to use the list feature for all entities, that is, the NLU will not attempt to extract any entities that are not present in the training set. This is thematically in-line with the nature of the Repository task where a chatbot answers questions about software repositories. In this context, an entity that does not exist in the repository (e.g., wrong Jira ticket number) is not useful for the chatbot and cannot be used to extract any information for the user. Then, using the NLUs' API, we define the entity types that exist in the Repository corpus, namely 1) Commit-hash 2) Jira Ticket 3) File Name, and use a fourth built-in entity type (Date Time).

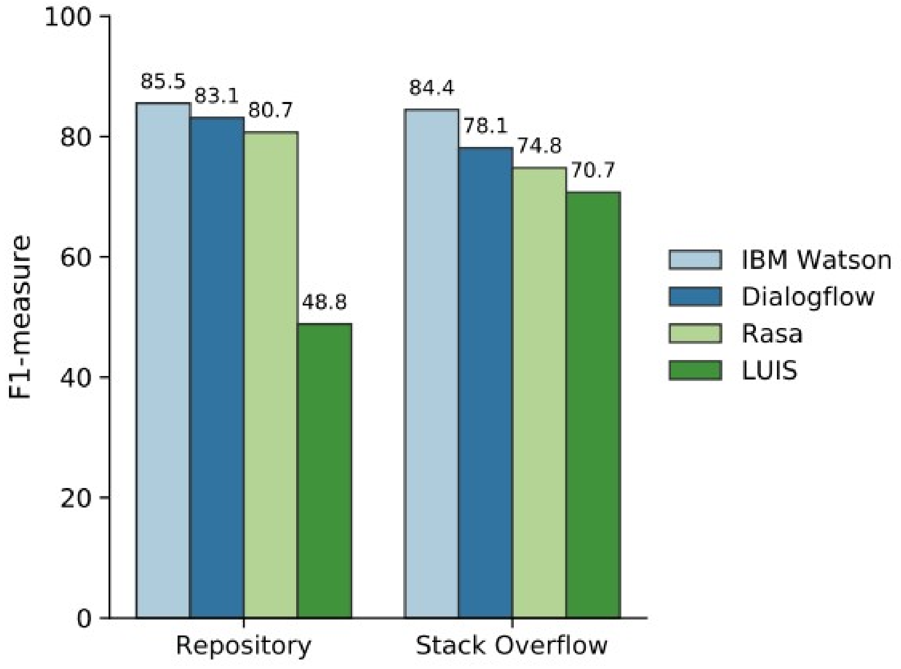


Fig. 3.2. Intent classification performance as F1-measure of the four NLUs.

CHAPTER 4

CASE STUDY RESULTS

In this section, we present the comparison of the NLUs' performance in terms of intents classification, confidence score, and entity extraction on the Repository and Stack-overflow tasks.

4.1 Intents Classification

To evaluate the NLUs in intents classification, we train and test each of the NLUs using the corpus from each of the two SE tasks. When testing the NLUs, we only consider the top scoring intent as the classified intent for two reasons. First, to emulate real life use-cases where chatbots use the intent with the highest corresponding score, as it is the intent with the highest probability of being correct. Second, to ensure that the evaluation of all NLUs is consistent, as Dialog flow only returns one intent with the corresponding confidence score in its response for a single query.

Results. Figure 2 shows the F1-measure for intent classification. The Figure presents the performance for each SE task, per NLU. The ranking is consistent across both tasks, showing that IBM Watson outperforms other NLUs, achieving an F1-measure of 85.5% in the Repository task and 84.4% in the Stack Overflow task. We also observe that for both SE tasks, LUIS comes last in intents classification.

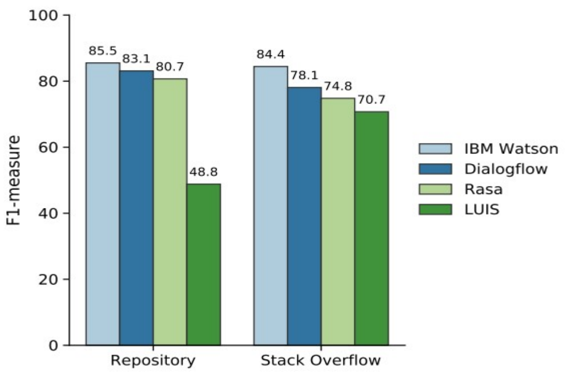


Fig. 4.1. Intent classification performance as F1-measure of the four NLUs.

4.2 NLUs Confidence scores

As discussed earlier in Section 2, every intent classification performed by the NLU has an associated confidence score. The confidence score is used to determine how much trust one can put into the intent classification of the NLU. Typically, NLUs should provide high confidence scores for intents that are correctly classified. For example, if a query is asking"What is the number of commits between 1-July- 2020 to 1-August-2020?" and the NLU provides a high confidence score (e.g., 0.98) when attributing the query to the *Count Commits By Date* intent, then the users of the NLU can trust these confidence scores. Also, the contrary is true, if an NLU provides high confidence scores to wrongly classified intents, then one would lose trust in the NLUs' produced confidence scores.

Results. Fig 4.2 shows the distributions of confidence scores returned by IBM Watson, Dialog flow, Rasa, and LUIS in the Repository and Stack Overflow tasks. From Figure 3, we observe that all NLUs return higher median confidence scores for the correctly classified intents compared to the incorrectly classified intents, for both tasks. The sole exception is Dialog flow, which has a higher median confidence score for incorrectly classified intents for the Repository task. Among the evaluated NLUs, Rasa stands out as being the NLU with the highest corresponding confidence scores medians (higher than 0.91) in both the Repository and Stack Overflow tasks, for the correctly classified intents. Furthermore, Rasa has the most compact distribution of confidence scores among other NLUs and the least overlapping confidence scores between the correctly classified and mis classified intents. This means that when Rasa returns a high confidence score, it is highly likely to be correct. To ensure that the difference in the confidence scores between the correctly and incorrectly classified intents across NLUs is statistically significant, we perform the non parametric unpaired Mann-Whitney U test on each NLU results. We find that the differences are statistically significant (i.e., p-value < 0.05) in all cases and for both, the Repository and Stack Overflow tasks, except for the results of Dialog flow in the Repository task. Generally, our results show that developers can trust the confidence score yielded by NLUs to assess if the NLUs have correctly classified the intent, or the chat-bot needs to trigger the fallback action, an action that is used when an NLU cannot determine a correct intent.

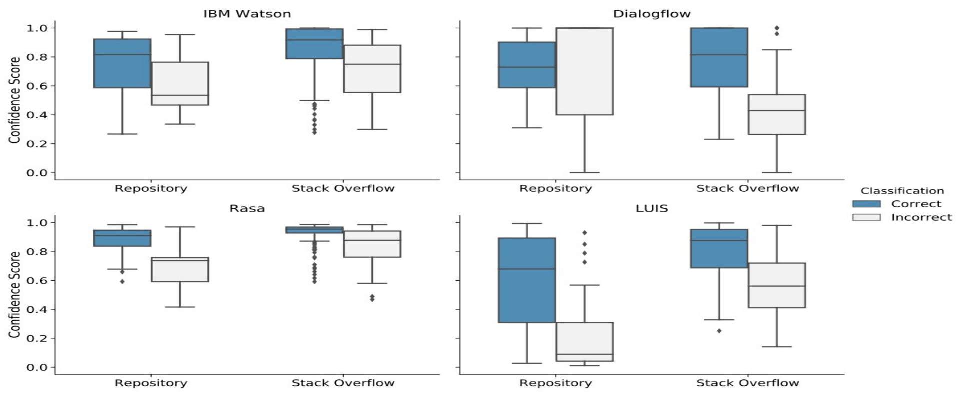


Fig. 4.2. Confidence score distribution for all NLUs and tasks.

4.3 Entity Extraction

To correctly answers users' queries, chatbots need to also correctly extract the relevant entities. We consider the extracted entity to be correct only if its type and value exactly match the type and value of expected entity for that query in the oracle. The reason behind our criteria is that extracting entities that are only partially correct (i.e., have only correct values or correct types), causes the chatbot to incorrectly reply to the user's query. Since there can exist a varying number and types of entities in each query, we need a mechanism to ensure that our evaluation accounts for such variance. To mitigate this issue, we calculate the precision, recall, and weighted F1-measure based on the number of entities in each entity type, for the entity extraction results. Results. Figure 4 presents the entity extraction results for the two tasks. To better interpret the results, we reiterate that extracting entities in the Repository and Stack Overflow tasks require different NLU features. For the Repository task, we configure the NLUs to extract entities using the list feature (i.e., list of synonyms) and entities are extracted as an exact match. In the Stack Overflow task, however, we configure the NLUs to use the prediction feature as discussed in Section 3.2, that is, requesting the NLUs to predict the correct entity. Given their differences, we discuss the results of each task separately, first describing the results for the Repository task and then the Stack Overflow task. Repository. According to Figure 4, LUIS is the best performing NLU with an average F1-measure of 93.7%, followed by Rasa (90.3%). Both IBM Watson and Dialog flow perform similarly (70.7%) when extracting entities in the Repository task. To better understand the factors influencing the performance, we examine the F1-measure of the NLUs in light of the different entity types. Table 6 shows the NLU's performance per entity type. We highlight, in bold, the best performing NLU for each entity type. Our results confifirm that LUIS is the best performing NLU when extracting the Commit-hash, Jira Ticket, and File Name entity types from the Repository task. IBM Watson also performs well in the Commit-hash and JiraTicket entity types, but it falls short when extracting File Name entities (F1-measure of 15.9%). This is due to IBM Watson's inability to differentiate between the normal words (e.g., 'To' and 'A') and entities of type File Name (e.g., 'To.java' and 'A.java'). In other words, it extracts entities based on their exact match with the training set without considering the entities' context when using the list feature . For Date Time entities, Rasa performs best in extracting all the dates correctly (F1-measure of 100%) from the given queries. Rasa uses a probabilistic context free grammar through their pipeline (Duckling) to extract the dates from the posed queries [61]. For other NLUs, we notice two reasons behind the incorrect extraction of the Date Time entities: 1) the misspelling of the date from the users (e.g., "what are the commits I submit on 27/5/2018 31/5/2018") 2) vague date formats in queries (e.g., "Tell me about the commits from 05-27 to 05-31 in 2018").

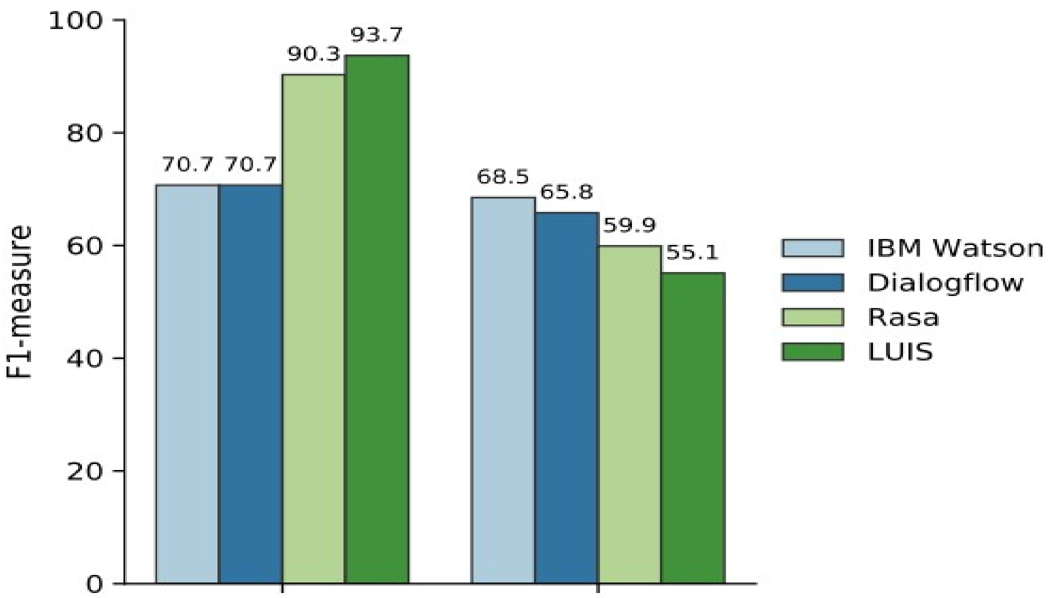
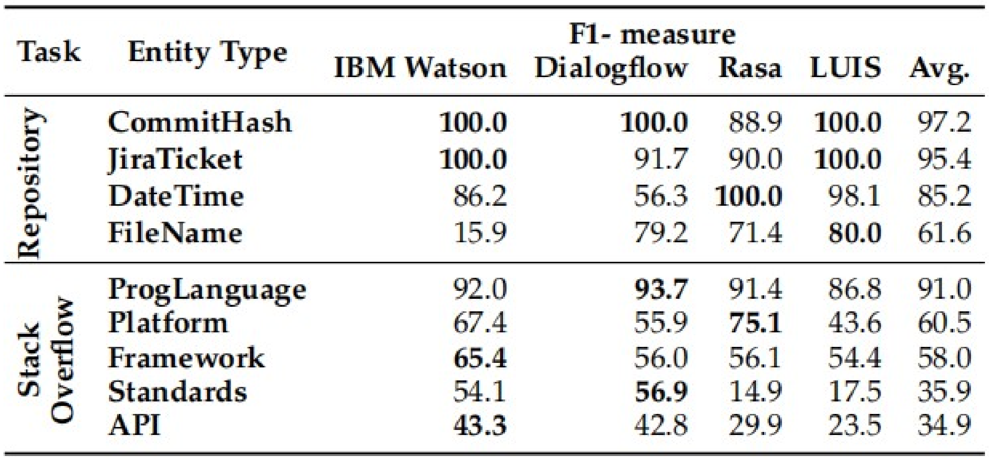


Fig. 4.3. Entity extraction performance as avg. F1-measure of the four NLUs.

Stack Overflow. We observe in Figure 4 that IBM Watson yields the best results (F1-measure of 68.5%), followed by Dialog flow (65.8%), Rasa (59.9%), and LUIS (55.1%).

Note that the performance of the NLUs on the Stack Overflow task is expectedly lower than the performance obtained in the Repository task, given that NLUs have to predict entities in a query. Similar to the case of the Repository task, we also examine the performance of the NLUs when extracting the different entity types in the Stack Overflow task. Table 6 shows that the performance of the NLUs varies from one entity type to the other, and that no NLU outperforms the rest in extracting the majority of entity types. Table 6 shows that, in the Stack Overflow task, both IBM Watson and Dialog flow outperform other NLUs in extracting two different entity types. Furthermore, we observe that some entity types are more difficult to extract than others. In particular, entities of type Framework, Standards, and API are difficult to extract (i.e., the four NLUs achieve an F1- measure < 60%). Upon closer investigation of the results, we find that more than 60% of the most difficult entities appear only once in the task, that is, they are unique entities. For example, the entity ‘DOM’ of type Standards in the query “How to tell if an element is before another element in DOM” is a unique entity as it occurs only once in the Stack Overflow task. This entity was not extracted by any of the evaluated NLUs. Consequently, NLUs tend to perform well when extracting entities which appear frequently in the training set (i.e., not unique).



Task	Entity Type	F1- measure				
		IBM Watson	Dialogflow	Rasa	LUIS	Avg.
Repository	CommitHash	**100.0**	**100.0**	88.9	**100.0**	97.2
	JiraTicket	**100.0**	91.7	90.0	**100.0**	95.4
	DateTime	86.2	56.3	**100.0**	98.1	85.2
	FileName	15.9	79.2	71.4	**80.0**	61.6
Stack Overflow	ProgLanguage	92.0	**93.7**	91.4	86.8	91.0
	Platform	67.4	55.9	**75.1**	43.6	60.5
	Framework	**65.4**	56.0	56.1	54.4	58.0
	Standards	54.1	**56.9**	14.9	17.5	35.9
	API	**43.3**	42.8	29.9	23.5	34.9

Table 6: Entity extraction performance as F1-measure per entity of the four NLUs.

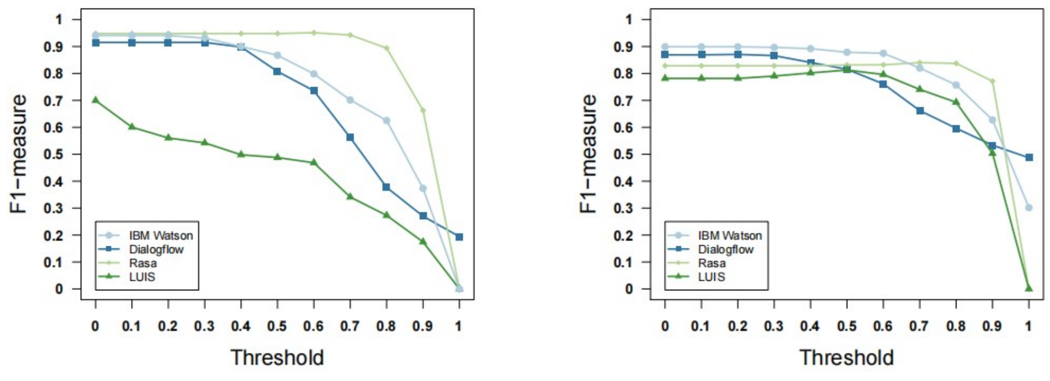
4.4 Unique Entities

The performance of the NLUs in extracting entities from the Stack Overflow task was affected by unique entities, as shown in Section 4.3. As the name suggests, unique entities appear just once in the datasets for the Stack Overflow task; thus, the NLUs have to predict their occurrences without prior training. It is important to note that there are no unique entities when evaluating the NLUs using the list feature be cause the NLUs have been trained on all entities that exist in the Repository task. To better understand the NLUs' ability to extract unique entities, we investigate the results from the Stack Overflow task, examining the NLU performance on queries containing only unique entities. Hence, queries containing any non-unique entity were excluded from this investigation. We find 58 queries that fit our criteria, and they include a total of 75 unique entities that are distributed, as shown in Table 8. Similarly to the evaluation conducted in Section 4.3, we calculate the precision, recall, and weighted F1-measure of the NLUs when extracting unique entities.

Table 7: NLUs' overall performance ranking.

NLU	Ranking in Repository Task			Ranking in Stack Overflow Task			Avg. Rank
	Intents Classification	Confidence Score	Entity Extraction	Intents Classification	Confidence Score	Entity Extraction	
IBM Watson	1	2	3*	1	2	1	1.7
Rasa	3	1	2	3	1	3	2.2
Dialogflow	2	3	3*	2	4	2	2.7
LUIS	4	4	1	4	3	4	3.3

* Same rank



(a) Ranking in Repository Task

(b) Stack Overflow Task

Figure 5: Analysis of theshold sensitivity in terms of F1-measure of the NLUs in the Repository and Stack Overflow tasks.

CHAPTER 5

CONCLUSION

Software chatbots are becoming popular in the SE community due to their benefits in saving development time and resources. An NLU lies at the heart of each chatbot to enable the understanding of the user's input. Selecting the best NLU to use for a chatbot that operates in the SE domain is a challenging task. In this paper, we evaluate the performance of four widely-used NLUs, namely IBM Watson, Google Dialog-flow, Rasa, and Microsoft LUIS. We assess the NLUs' performance in intents classification, confidence score, and entity extraction using two different tasks designed from a Repository and Stack Overflow contexts.When considering all three aspects (intents classification,confidence scores, and entity extraction), we find that IBM Watson is the best performing NLU for the studied SEtasks. For each individual aspect, in intents classification,IBM Watson outperforms other NLUs for both tasks. On the other hand, when it comes to confidence scores, all NLUs(except for Dialog flow) return high confidence scores for the correctly classified intents. Also, we find that Rasa is the most trustworthy NLU in terms of confidence score.Finally, LUIS and IBM Watson achieve the best results in extracting entities from the Repository and Stack Overflow tasks, respectively. Moreover, our results shed light on the characteristics that affect the NLUs' performance in intents classification (e.g., # Training Samples) and entity extraction(e.g., unique entities). Therefore, we encourage researchers to develop techniques and methods that enhance the NLUs' performance for tasks with different characteristics. We be live that our work guides chatbot practitioners to select the NLU that best fits the SE task performed by their chatbots.Our study paves the way for future work in this area.First, our results show that NLUs tend to perform well when they are trained on more examples. Therefore, we plan to examine different datasets augmentation techniques to generate more training examples for the NLUs to enhance their performance. Also, we believe that there is a need for more studies that compare different NLUs using more datasets to benchmark NLUs in the SE context. We contribute towards this effort by making our datasets publicly available .

REFERENCES

[1] M.-A. Storey and A. Zagalsky, “Disrupting developer productivity one bot at a time,” in Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ser. FSE 2016.ACM, 2016, pp. 928–931.

[2] B. Xu, Z. Xing, X. Xia, and D. Lo, “Answer Bot: Au tomated generation of answer summary to developers technical questions,” in Proceedings of the 32NdIEEE/ACM International Conference on Automated Soft ware Engineering, ser. ASE 2017. Piscataway, NJ, USA:IEEE Press, 2017, pp. 706–716.

[3] A. Abdellatif, K. Badran, and E. Shihab, “MSRBot: Using bots to answer questions from software repositories,” Empirical Software Engineering (EMSE), p. To Appear, 2019.