# FACER-Unity: An API based code recommendation tool for Unity scripting

Aena Hassan
*School of Science and Engineering*
*Lahore University of Management Sciences*
Lahore, Pakistan
20030003@lums.edu.pk

Taha Muzammil
*School of Science and Engineering*
*Lahore University of Management Sciences*
Lahore, Pakistan
20030024@lums.edu.pk

***Abstract*—Developers spend significant amount of time to search and understand code examples that implement their desired software features. To solve this issue we are implementing code search and recommendation within working IDE. Existing code search techniques typically focus on finding code snippets for a single given query, which means that developers need to perform a separate search for each desired functionality. In this project we propose FACER plugin for Unity 3d game development in C# language.The plugin would suggest code for additional features.**

***Index Terms*—Code search, code recommendation, Code reuse, Unity Game Development, API usage, parse tree.**

## I. Introduction

In recent years, there is a huge rise in game industry and Unity 3D is one of the most used tool to develop 2D and 3D games. According to the survey [1], Unity 3D Engine has a global market share of Game Engine which is around 45%, while 47% of game developers prefer Unity as a smart game development tool. For any help during code programmers prefer google for searching. Searching on google is very time consuming because google returns lot of search results from which most of the results are not useful for developer. To tackle this issue we want to integrate code search and recommendations within our environment to help programmers in searching of relevant code easily.

For example, if Amy is developing a shooter game one of the functionalities might be shoot bullet. Then she might search for "shoot bullet" and use the returned snippet in her code. Then she might to implement "update bullets", "play collision sound" etc. These are all functionalities related to the original code. The challenge is to provide recommendations for these functionalities without the user having to explicitly perform a search for each desired functionality.

Programmers iteratively add features by reusing code examples [2]. According to StackOverflow 2021 survey $27.86\%$ developers use C# [3]. To speed up development, developers often resort to code search to find code that they can reuse for certain features in their application [5], [6]. Most of the existing code search systems focus on providing code corresponding to a single query related to the current feature the developer needs to implement [7]–[14]. As a result, developers may have to conduct a new search for every next feature they need to implement and later integrate the obtained code. Existing code search and recommendation systems do not support the need to find code for additional features related to a developer's query. On the other hand, existing feature recommendation systems enable the exploration of text-based related features and either support domain analysis for the requirements gathering phase [15]–[19] or enable rapid prototyping by recommending related code modules [20]. However, they do not provide code examples at a fine-grained method-level for reuse. Hence, we identify two gaps in existing systems: one is the lack of support for providing related code recommendations in existing code search systems and the other is the inability of existing feature recommendation systems to provide code for features at the method-level granularity. [24]–[26] discuss systems that provide support for $C\#$ language.
In Unity(C#) development tasks example code is often required to speed up the process. In this project we will provide a FACER-Unity a Unity (C#) plugin to provide relevant code against queries. This plugin will use existing FACER as its backend [4].

FACER generates related method recommendations in two stages. The first stage corresponds to traditional code search where any existing code search technique [7], [9], [10], [12], [14]. Given a developer's feature query in the form of natural language description, the search stage of FACER recommends a set of methods that implement the desired feature. Upon selection of one of these recommended methods by the developer, the second stage of FACER starts. In this second stage, FACER provides subsequent recommendation of related methods for reuse.

The concept of FACER is not for any particular language and its application. FACER can be mold to any programming language, In our work we are using FACER for C# language which is used for 2D and 3D game development in Unity. FACER not only recommends code but also recommends additional related methods, such as include functionality for resizing a bitmap, getting a URI to save the cropped image,

getting the URI to an image received from a capture by camera, and also decoding an image from a URI. Through these recommendations developer can obtain information about other related features to enhance the application.

## II. Related Work

There are various systems in the literature that enable code search, code reuse and also recommend code. Here we discuss systems that are available for the $C\#$. language.

Jiao et al., suggests a a question-driven source code recommendation service based on the source code in stack overflow. [24] This paper uses a deep learning approach to match the question and the source code available on stack overflow. The user can select the tag to check the domain of the query. The system then recommends source code based on the query. This system is different from our proposed system FACER-Unity, as it only recommends code based on user query and does not recommend snippets that may be used later in the development process.

Sando [25] is a similar tool that supports $C\#$ language. The developer can issue query using a search box. The system recommends code from the developers local code base. It assumes that the developer is not familiar with the local code base. This tool has the limitation that it only recommends code from the local code base and does not recommend new code. This system only supports code search and does not provide recommendations for future use.

Mentor [26] is a code recommendation system that uses prediction by partial matching to match a query with the code base. This system is designed for environments where version control systems are used. It assumes that similar change requests have similar solutions. It supports multiple programming languages and also works for change request descriptions that are in English or Portuguese. Many implementations of such systems are also found in JAVA language.

Many implementations of such systems are available for JAVA language. Keivanloo et al. [7] proposed a system that retrieves code examples from a corpus of code snippets based on free-form querying (composed of keywords). They create a p-string for each line of a code snippet in the corpus and encode matching p-strings as a pattern. By applying identifier splitting techniques on all strings that belong to an encoded pattern, they extract a set of associated keywords. The retrieval involves matching keywords of a user query with keywords associated with patterns and getting the most popular code examples containing the matched patterns.

In this paper [8], they discussed an approach where they first locate similar code on the basis of similarity given a user query and then recommends JAVA packages. The main difference from FACER is that it provides recommendations on the package level while FACER recommends related methods.

Sniff [9] is a system that uses library documentation to annotate code with natural language. It then provides recommendations based on the similarity between the query and the annotation.

Moreover, work has also been done using deep learning techniques. There new design [21] is the development of API recommendation system for Android programming using Auto-encoder Neural Network based model and validates the performance of the model through experiments.

H. Cho et al. [23] worked on implementing code recommendation using RNN based on Interaction History. This technique worked perfectly if input data does not exactly match with learned data because this was the problem of statistical language model technique and gained average precision of 91% and an average recall of 71%.

In [22] Eiji Adachi Barbosa et al. worked on recommendation system for exception handling code. They discuss implementing and assessing a recommendation system for recommending code fragments implementing exception handling code. Instead of using these fragment for developers, these are meant to be used by the developers as examples of how to possibly handle their exceptions.

## III. Objectives

In this paper we aim to implement and extend existing FACER tool to recommend code snippets in C#. We aim to compile a code base using open source GITHUB projects. Through this dataset we first parse the C# code for program analyzer. We will compile a code fact repository using existing FACER methods. The extension will predict code snippets based on the initial query. First we will generate recommendations and the next step would be to make a plugin for Unity IDE (Visual Studio).

## IV. Methodology

In this project we will develop FACER-Unity an API usage-based code recommendation tool for Unity developers. We will first compile a unity codebase using open source GitHub projects. The next step would be to integrate it with the existing tool FACER [4] to detect unique features based on similar API usages. Then it recommends similar features using mined patterns in the codebase.

### A. Dataset

For our project we have collected dataset related to game development (Unity) in $C\#$. We collected projects of three different categories: (1) Shooting Games, (2) Puzzle Games, and (3) Endless Runner Games. From these categories we selected total 78 projects from which 30 are shooting games,

25 projects are of puzzle games, and 23 projects are from endless runner games. We choose multiple applications from each category to discover the frequently co-occurring features across similar category applications.

For the dataset we used GitHub projects by searching category *name* as a prefix and postfixed with *Unity* in GitHub search. After that we select C# language from the dropdown filter and sort the projects on the basis of stars, and check the hashtags of each project whether it has #Unity3d, #Unity, or #Unity2d then pick top projects which has hashtag and in C# language. 'Fig. 1" shows the distribution of dataset.

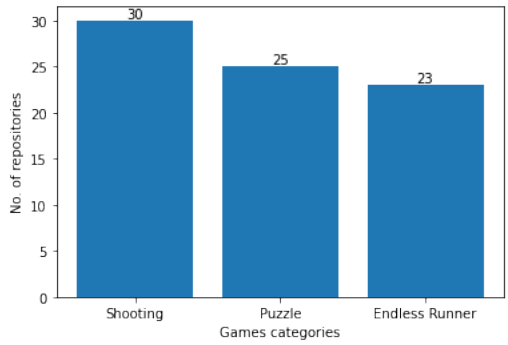


Fig. 1. The number of GitHub repositories from the three categories

### B. Parsing Approach

For code recommendation, first task is to do parsing of dataset. For parsing we are using ANTLR (ANother Tool for Language Recognition) tool to extract Method Calls, API Calls, and Keywords. ANTLR is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It is widely used to build languages, tools, and frameworks. From a grammar, ANTLR generates a parser that can build and walk parse trees.

For the setup of ANTLR there are few pre-requisites required. ANTLR is written in java so first of all we installed latest java version, download and install ANTLR from ANTLR website [27]. ANTLR has an IntelliJ plugin, so download Rider from the JetBrains website [28]. Next step is to install ANTLR plugin to use it in Rider, for the installation of plugin go to File > Settings > Pluigns then search for ANTLR an install the ANTLR v4 plugin.

For parsing we first define a grammar on the basis of the grammar the parser generates an abstract syntax tree.Using the abstract syntax trees we identify all the calls. For each call we identify the line number in the method body.

## REFERENCES


[1] Blackman, Sue. Beginning 3D Game Development with Unity 4: All-in-one, multi-platform game development. Apress, 2013.

[2] Joel Brandt, Philip J Guo, Joel Lewenstein, Mira Dontcheva, and Scott R Klemmer. Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code. In Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, pages 1589–1598, 2009.

[3] Stack overflow developer survey 2021. Stack Overflow. (n.d.). Retrieved February 5, 2022, from https://insights.stackoverflow.com/survey/2021#technology

[4] Abid, S., Shamail, S., Basit, H.A. et al. FACER: An API usage-based code-example recommender for opportunistic reuse. Empir Software Eng 26, 110 (2021). https://doi.org/10.1007/s10664-021-10000-w

[5] Sadowski C, Stolee KT, Elbaum S (2015) How developers search for code: a case study. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. ACM, pp 191–201

[6] Xia X, Bao L, Lo D, Kochhar PS, Hassan AE, Xing Z (2017) What do developers search for on the web? Empir Softw Eng 22(6):3149–3185

[7] Keivanloo, I., Rilling, J., & Zou, Y. (2014). Spotting working code examples. In Proceedings of the 36th International Conference on Software Engineering. ICSE '14: 36th International Conference on Software Engineering. ACM. https://doi.org/10.1145/2568225.2568292

[8] McMillan C, Grechanik M, Poshyvanyk D, Xie Q, Fu C (2011) Portfolio: finding relevant functions and their usage. In: Proceedings of the 33rd International Conference on Software Engineering. ACM, pp 111–120

[9] Bajracharya SK, Ossher J, Lopes CV (2010) Leveraging usage similarity for effective retrieval of examples in code repositories. In: Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering. ACM, pp 157–166

[10] Chatterjee, S., Juvekar, S., Sen, K. (2009). SNIFF: A Search Engine for Java Using Free-Form Queries. In Fundamental Approaches to Software Engineering (pp. 385–400). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-00593-0-26

[11] Ishihara T, Hotta K, Higo Y, Kusumoto S (2013) Reusing reused code. In: 2013 20th working conference on Reverse engineering (WCRE). IEEE, pp 457–461

[12] Gu X, Zhang H, Kim S (2018) Deep code search. In: 2018 IEEE/ACM 40Th international conference on software engineering (ICSE). IEEE, pp 933–944

[13] Lv F, Zhang H, Lou J-g, Wang S, Zhang D, Zhao J (2015) Codehow: Effective code search based on api understanding and extended boolean model (e). In: 2015 30Th IEEE/ACM international conference on automated software engineering (ASE). IEEE, pp 260–270

[14] Sachdev S, Li H, Luan S, Kim S, Sen K, Chandra S (2018) Retrieval on source code: A neural code search. In: Proceedings of the 2Nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, MAPL 2018. ACM, New York, pp 31–41. ISBN 978-1-4503-5834-7. https://doi.org/10.1145/3211346.3211353

[15] He J, Zhang J, Li X, Ren Z, Lo D, Wu X, Luo Z (2019) Recommending new features from mobile app descriptions. ACM Trans Softw Eng Methodol (TOSEM) 28(4):22

[16] Chen X, Zou Q, Fan B, Zheng Z, Luo X (2018) Recommending software features for mobile applications based on user interface comparison. Requir Eng:1–15

[17] Hong Y, Lian Y, Yang S, Tian L, Zhao X (2016) Recommending features of mobile applications for developer. In: International conference on advanced data mining and applications. Springer, pp 361–373

[18] Yu Y, Wang H, Yin G, Bo L (2013) Mining and recommending software features across multiple web repositories. In: Proceedings of the 5th Asia-Pacific Symposium on Internetware. ACM, pp 9

[19] Dumitru H, Gibiec M, Hariri N, Cleland-Huang J, Mobasher B, Castro-Herrera C, Mirakhorli M (2011) Ondemand feature recommendations derived from mining public product descriptions. In: Proceedings of the 33rd International Conference on Software Engineering. ACM, pp 181–190

[20] McMillan C, Hariri N, Poshyvanyk D, Cleland-Huang J, Mobasher B (2012) Recommending source code for use in rapid software prototypes. In: Proceedings of the 34th International Conference on Software Engineering. IEEE Press, pp 848–858

[21] J. Liu, Y. Qiu, Z. Ma and Z. Wu, "Autoencoder based API Recommendation System for Android Programming," 2019 14th International Conference on Computer Science & Education (ICCSE), 2019, pp. 273-277, doi: 10.1109/ICCSE.2019.8845349.

[22] E. A. Barbosa, A. Garcia and M. Mezini, "A recommendation system for exception handling code," 2012 5th International

Workshop on Exception Handling (WEH), 2012, pp. 52-54, doi: 10.1109/WEH.2012.6226601.

[23] H. Cho, S. Lee, and S. Kang, “A Code Recommendation Method Using RNN Based on Interaction History,” KIPS Transactions on Software and Data Engineering, vol. 7, no. 12, pp. 461–468, Dec. 2018.

[24] H. Yin, Z. Sun, Y. Sun, and W. Jiao, “A Question-Driven Source Code Recommendation Service Based on Stack Overflow,” 2019 IEEE World Congress on Services (SERVICES). IEEE, Jul. 2019. doi: 10.1109/services.2019.00102.

[25] X. Ge, D. Shepherd, K. Damevski, and E. Murphy-Hill, “How the Sando search tool recommends queries,” 2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE). IEEE, Feb. 2014. doi: 10.1109/csmr-wcre.2014.6747210.

[26] Y. Malheiros, A. Moraes, C. Trindade, and S. Meira, “A Source Code Recommender System to Support Newcomers,” 2012 IEEE 36th Annual Computer Software and Applications Conference. IEEE, Jul. 2012. doi: 10.1109/compsac.2012.11.

[27] ”ANTLR”, Antlr.org, 2022. [Online]. Available: https://www.antlr.org/. [Accessed: 08- Mar- 2022]

[28] ANTLR v4 - IntelliJ IDEs Plugin — Marketplace”, JetBrains Marketplace, 2022. [Online]. Available: https://plugins.jetbrains.com/plugin/7358-antlr-v4. [Accessed: 08- Mar- 2022]