Bing Developer Assistant: Improving Developer Productivity by Recommending Sample Code

Hongyu Zhang[†], Anuj Jain[§], Gaurav Khandelwal[§], Chandrashekhar Kaushik[§], Scott Ge[T], Wenxiang Hu[†]

[†] Microsoft Research, Beijing 100080, China, China

[§]Microsoft Corporation, Hyderabad, India, [T]Microsoft Corporation, Redmond, USA, USA

{honzhang, anjai, gaurav.khandelwal, chkaus, jialiang.ge, v-wenxhu}@microsoft.com

ABSTRACT

In programming practice, developers often need sample code in order to learn how to solve a programming-related problem. For example, how to reuse an Application Programming Interface (API) of a large-scale software library and how to implement a certain functionality. We believe that previously written code can help developers understand how others addressed the similar problems and can help them write new programs. We develop a tool called Bing Developer Assistant (BDA), which improves developer productivity by recommending sample code mined from public software repositories (such as GitHub) and web pages (such as Stack Overflow). BDA can automatically mine code snippets that implement an API or answer a code search query. It has been implemented as a free-downloadable extension of Microsoft Visual Studio and has received more than 670K downloads since its initial release in December 2014. BDA is publicly available at: *http://aka.ms/devassistant*.

CCS Concepts

• **Software and its engineering→Reusability**

Keywords

API; API Usage Extraction; Code Search; GitHub; Software Reuse

1. INTRODUCTION

Programming is sometimes hard. In practice developers often face many challenges that are associated with problem solving. For example, developers often wonder how to implement a certain functionality (e.g., *how to save an image in PNG format in C#*). Although APIs (Application Programming Interfaces, such as *Image.Save* and *Image.Tag* in .NET framework) provide an important form of software reuse and have been widely used for effective problem solving, developers still need to know *how to reuse an API?* A large-scale software framework, such as the .NET framework, could contain hundreds or even thousands of APIs. Programmers often do not remember exactly how a certain API function should be reused. Many APIs are not well documented either. In a survey conducted at Microsoft in 2009, 67.6% respondents mentioned that there are obstacles caused by inadequate or absent resources for learning APIs [9, 10]. Besides API functions, developers may also need to know the usage of a public class, structure, or property. Therefore, it is desirable if an IDE can provide developers with sample code about APIs whenever they encounter programming difficulties.

FSE'16, November 13–18, 2016, Seattle, WA, USA
© 2016 ACM. 978-1-4503-4218-6/16/11...$15.00
http://dx.doi.org/10.1145/2950290.2983955

Over the years, millions of software programs have been developed and deployed. The source code of these programs is typically stored in software repositories (such as GitHub) and can be treated as important reusable assets for developers. Previously written programs can help developers understand how others addressed the similar problems and can serve as a basis for writing new programs. Furthermore, the emergence of online developer-centric question answering (QA) forums such as Stack Overflow also provide abundant resources (including sample code) for solving programming-related problems.

Many modern Integrated Development Environments (IDEs, such as Microsoft Visual Studio) provide rich features for effective program development. However, most of the IDEs do not provide sample code to developers. Although developers can turn to a general web search engine such as Bing or Google to look for a sample solution, they often need to leave their IDE environment frequently and switch the context between IDE and web browser. This could result in productivity loss. Also, some programming-related problems such as the usage of a certain API, may not be well supported by a general web search engine.

We develop a tool, called Bing Developer Assistant (BDA), to help improve developer productivity by mining software repositories and web pages. We deploy the backend of BDA as a Microsoft Azure service and implement the frontend as a Visual Studio extension. BDA can automatically detect the APIs under editing in the current programming context via the IntelliSense feature of Visual Studio and provide sample code that contains the usage of the APIs. BDA also leverages Bing search engine to support free-text based code search queries. BDA provides an in-place entry point to sample solutions for more efficient task completion.

Currently, BDA supports C/C++/C# languages. BDA has been well received by developers. It has attracted more than 670K downloads since its initial release in December 2014. BDA is now available for download at: *http://aka.ms/devassistant.* We are also extending BDA as a service to support more languages such as Java and JavaScript and more application scenarios. In this paper, we will describe the development of BDA and the results.

2. GETTING SAMPLE CODE FROM SOFTWARE REPOSITORIES

In this section, we describe the proposed framework for extracting API sample code from a software repository. The sample code is for understanding the usage of a variety of APIs including public Class, Function, Structure, Property, etc. Figure 1 shows an overall framework of our approach. We first construct a large-scale software repository by crawling source code projects from websites such as MSDN and GitHub. The repository contains source files written in different languages such as C/C++/C#/Java. We then utilize compilers such as Clang[1] (for C/C++), Roslyn[2] (for C#), and JDT[3] (for Java) to statically analyze these source files and produce an intermediate representation (Abstract Syntax Trees). The API-related source code information can be inferred from the ASTs. The resulted API usage data (including information about project, file, line, etc.) is stored in Azure Table[4]. Finally, the associated sample code can be extracted for use (by clients such as Visual Studio IDE).

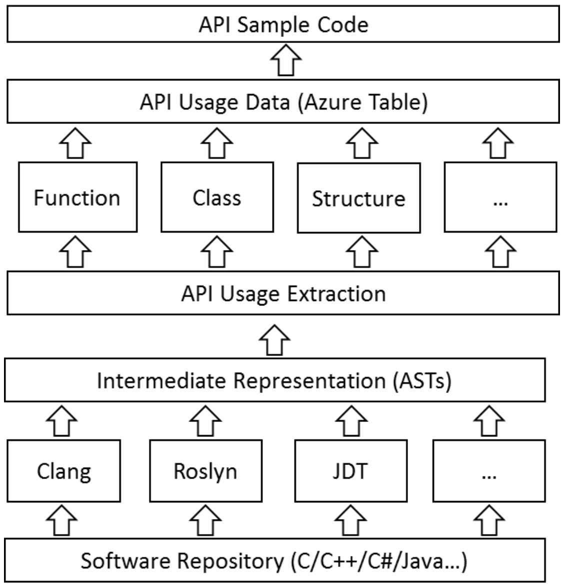

API Sample Code
API Usage Data (Azure Table)
Function
Class
Structure
...
API Usage Extraction
Intermediate Representation (ASTs)
Clang
Roslyn
JDT
...
Software Repository (C/C++/C#/Java...)

Figure 1: Mining API usage from software repositories

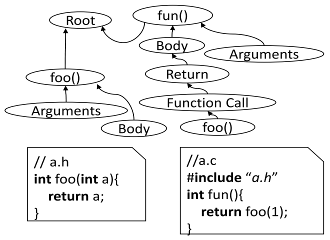
In particular, we develop a static analyzer, which parses the source code file into ASTs and then performs data extraction upon visiting each AST node. The analyzer is based on a compiler frontend (such as Clang's *LibTooling,* which is a library for exposing ASTs). In this way, we do not require any .dll/.so files or any specific runtime environment.

The algorithm used by the static analyzer is shown in Figure 2. It contains three main parts: the first one (*Main*) is to parse the source code file to AST, the second one (*Visit*) is to traverse the nodes in the AST, and the third one (*Process_**) is a set of functions corresponding to different types of AST nodes. Given a source code file and the directories where the header files should be, the analyzer parses the source code file and the associated header files. The AST for the source code file and the ASTs for the header files are combined into one large AST as an intermediate representation. After the combined AST is generated, the analyzer performs an in-order traversal of the AST. When the analyzer visits the nodes that we are interested in, such as function call or variable declaration, it further processes the AST node by its type. In Figure 2, we only show the pseudocode for processing the function calls. We also implement a series of processing methods for other API types such as Classes and Objects.

Algorithm 1 Analysis of C/C++ files

```
S ← One Source File
H ← Header files
AST ← ∅

function MAIN
    Sourcefile ← S
    Headfiles ← H
    AST ← PARSER(Sourcefile, Headfiles)
    for Node ∈ AST root's Children do
        VISIT(Node)
    end for
end function

function VISIT(Node)
    if Node ∈ S then
        switch Node do
            case FunctionCall
                PROCESS_FUNCALL(Node)
            case VariableDeclaration
                PROCESS_VARDECL(Node)
            case Otherkindsofnodes
                ...
        for Child ∈ Node's Children do
            VISIT(Child)
        end for
    end if
end function

function PROCESS_FUNCALL(Node)
    Callee ← EXTRACT_CALLEE(Node)
    Decl ← FIND_DECLARATION(Callee)
    if Decl ≠ ∅ then
        File = WHERE_IS_FROM(Decl)
        if File ≠ S then
            LibName =WHICH_LIBRARY(File)
            LocationInfo = GET_LOCATIONINFO(Node)
            SAVE_DATA(Callee, LocationInfo, LibName)
        end if
    end if
end function
```

Figure 2: The algorithm for extracting API usage data



```
// a.h
int foo(int a){
   return a;
}
```

```
//a.c
#include "a.h"
int fun(){
   return foo(1);
}
```

Figure 3: An example of API call

We illustrate the algorithm using an example of an API call (Figure 3). Figure 3 contains a fragment of the whole AST, as well as two corresponding source code files. When we visit the nodes of "*a.c*",

[1] http://clang.llvm.org

[2] https://github.com/dotnet/roslyn

[3] http://www.eclipse.org/jdt

[4] https://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-tables/

we find that a node is actually a function call – "*foo(1)*". We use the callee's identifier to look up its valid declaration in the header file's AST nodes. As a result, we find that the declaration node in the "*a.h*" is what we want. We then check where this file is from. In this case, the "*a.h*" file is from the same directory as the "*a.c*", therefore we mark it as a user-defined API and record this along with its source code location in "*a.c*".

One challenge we faced is about third-party library support. This is actually reflected by the *Which_Library* function in Figure 2. We need to know which library an API *foo* comes from. If we are able to solve the dependency problem (e.g., in the case of C#, whose dependency can be resolved with the help of Nuget[5]), we build a list of <*library name, library's API list*> pairs. After parsing a source code file to AST and visiting each node in the AST, we can extract the APIs we want to know and search for the corresponding library name.

However, in some languages such as C/C++, we often cannot build a project successfully by executing the *make* file because the platform/environmental settings of the project. It is also not easy to resolve all the project dependencies. The absence of some third-party libraries makes it even more challenging to identify an API of a third-party library as the associated header files are missing. Considering the situation described in Figure 3, if the header file "*a.h*" is missing, we only know that *foo* is a function call statement but we do not know where the function is declared. In this case, we would fail to compile the source code file without the method.

BDA supports a list of commonly-used third-party libraries. We use Nuget[5] and GitHub to select the most popular third-party libraries (e.g., we select 1200 popular libraries for C/C++) that are widely used by developer community. We also generate a list of <*path, libname*> pairs, where *libname* indicates the name of a third-party library and *path* indicates the exact path to the header file. During program analysis, we obtain the path of the API under analysis and search it in the list of <*path, libname*> pairs. In this way, we know the third-party library the API belongs to, even we cannot build the project successfully.

After processing all the projects in software repository, we obtain the API usage data (including information about the API name, the library it belongs to, project, file, line number, etc.). We store these data in Microsoft Azure Table. Based on the usage data, sample code for the APIs can be readily retrieved for various application scenarios.

At the BDA client side, we hook the IntelliSense[6] interface of Visual Studio to obtain the context of program under editing. The IntelliSense interface can automatically provide developers with a list of APIs that could be used in the current programming context. Such API information is used to trigger our BDA sample code service within the Visual Studio environment.

3. GETTING SOLUTIONS FROM BING SEARCH ENGINE

To help developers understand how to implement a certain functionality, we integrate a code search component into Developer Assistant. Developers can enter a natural language query (such as *how to save an image*) describing the problem at hand. BDA can automatically invoke a web search using the Bing search engine (www.bing.com) and suggest relevant snippets obtained from the returned web pages (such as Stack Overflow and MSDN web pages). Figure 4 shows an overall framework of mining sample code using Bing.

More specifically, given a user query, BDA first extends the original query by appending the programing language used in the current programming context (such as "*how to save an image in c#*"). It then invokes the Bing search engine to find web pages related to that query. Upon retrieving the top 10 web pages, BDA identifies and extracts code from the HTML pages (based on HTML tags such as *hprei, hcodei, hpi*, and *hdivi*). BDA also ranks the extracted snippets based on relevancy and the number of votes/accepts, and displays them within the Visual Studio environment. In this way, users do not have to launch a separate web browser to perform a code search query. Our user study shows that the average response time per query is 1.5 second, which provides a smooth user experience. More details about the code search feature of BDA can be found at [17].

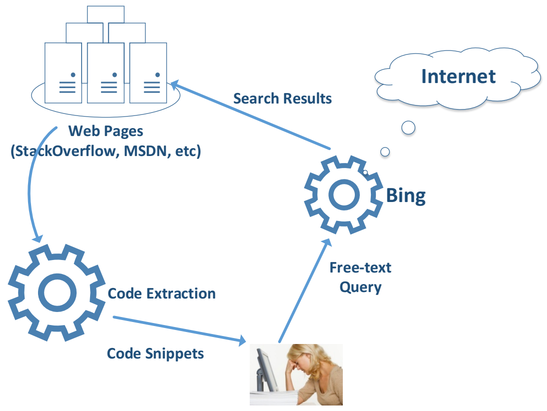


Figure 4: Mining sample code using Bing

Apart from providing code snippets, BDA also supports the recommendation of sample projects based on a user query. Code sample projects are complete Visual Studio solutions that developers can download, build and run. Upon receiving a free-text user query, BDA can invoke Bing search engine to retrieve most relevant projects from sites like MSDN and GitHub. The project information (such as project URL, author, license, ratings, etc.) are extracted and displayed within the Visual Studio environment. Developer productivity could be improved by reusing the entire solution.

BDA can also provide solutions for compilation errors. Once a compilation error occurs when compiling a program using Visual Studio, the user can simply right click and invoke contextual search. The BDA client will extract the current context (e.g. error codes/message, project type, data types, programming language being used, etc.). It then sends the context to the backend service, which will in turn extract relevant features from the context, generate a contextual query, and invoke Bing search engine to obtain the sample solutions for the compilation error. The obtained sample solutions are returned to the user. As an example, say we are editing a C++ program and using "*std::string*" class to store a "hello world" string. If we forget to include the header file "*string*", Visual Studio will report an error message such as *'string' is not a member of 'std'*. This message could be a bit confusing to some new developers. BDA can help with it by providing a sample solution. The error message and other contextual information are automatically combined to form a query, which is passed to BDA

[5] https://www.nuget.org/

[6] https://msdn.microsoft.com/en-us/library/hcw1s69b.aspx

backend service for Bing search. The Bing search results (in this case, a Stack Overflow webpage about solving the same compilation problem) is returned and displayed within Visual Studio.

4. TOOL IMPLEMENTATION AND RESULTS

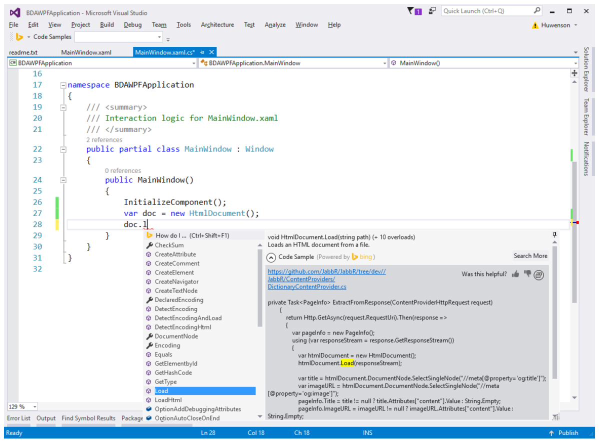
4.1 Tool Implementation

To obtain sample code about API usage from source code repositories, we construct a large-scale codebase by crawling projects from MSDN and GitHub. In order to remove dummy projects, for GitHub we only crawl projects that have at least 1 star. In total, we have crawled 65,253 projects. The total size of these projects is around 437 GB, containing about 3.5 million source code files. Table 1 shows the statistics of the obtained results.

Table 1. The statistics of the obtained API usage data

Language	#Projects	#Files	#Unique APIs	#Code Snippets
C/C++	12,162	730,084	2,980,543	43,944,234
C#	26,322	907,632	698,651	12,336,251
JavaScript	17,115	453,449	21,280	4,205,998
Java	9,654	1,398,695	68,958	17,679,077
Total	65,253	3,489,860	3,769,432	78,165,560

The API usage data and sample code are stored in a number of Microsoft Azure servers located around the world. The indexing and retrieving process is performed by Microsoft Azure Table. The client side is an extension that supports Microsoft Visual Studio (2012-2016). We hook the IntelliSense[7] interface of Visual Studio to obtain the context of program under editing. The APIs provided by IntelliSense trigger the backend BDA service. The returned sample code about the APIs are automatically obtained from Azure servers and displayed within Visual Studio.


BDAWPFApplication - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Code Samples
readme.txt MainWindow.xaml MainWindow.xaml.cs*
BDAWPFApplication BDAWPFApplication.MainWindow MainWindow()
How do I ... (Ctrl+Shift+F1)
CheckSum
CreateAttribute
CreateComment
CreateElement
CreateNavigator
CreateTextNode
DeclaredEncoding
DetectEncoding
DetectEncodingAndLoad
DetectEncodingHtml
DocumentNode
Encoding
Equals
GetElementbyId
GetHashCode
GetType
Load
LoadHtml
OptionAddDebuggingAttributes
OptionAutoCloseOnEnd
Code Sample (Powered by bing)
Search More
Was this helpful?
Ln 28 Col 18 Ch 18 INS
Publish

```
namespace BDAWPFApplication
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    2 references
    public partial class MainWindow : Window
    {
        0 references
        public MainWindow()
        {
            InitializeComponent();
            var doc = new HtmlDocument();
            doc.l
        }
    }
}
```

void HtmlDocument.Load(string path) (+ 10 overloads)
Loads an HTML document from a file.

https://github.com/JabbR/JabbR/tree/dev//JabbR/ContentProviders/DictionaryContentProvider.cs

```
private Task<PageInfo> ExtractFromResponse(ContentProviderHttpRequest request)
    {
        return Http.GetAsync(request.RequestUri).Then(response =>
        {
            var pageInfo = new PageInfo();
            using (var responseStream = response.GetResponseStream())
            {
                var htmlDocument = new HtmlDocument();
                htmlDocument.Load(responseStream);

                var title = htmlDocument.DocumentNode.SelectSingleNode("//meta[@property='og:title']");
                var imageURL = htmlDocument.DocumentNode.SelectSingleNode("//meta
[@property='og:image']");
                pageInfo.Title = title != null ? title.Attributes["content"].Value : String.Empty;
                pageInfo.ImageURL = imageURL != null ? imageURL.Attributes["content"].Value :
String.Empty;
```

Figure 5: Recommending API sample code

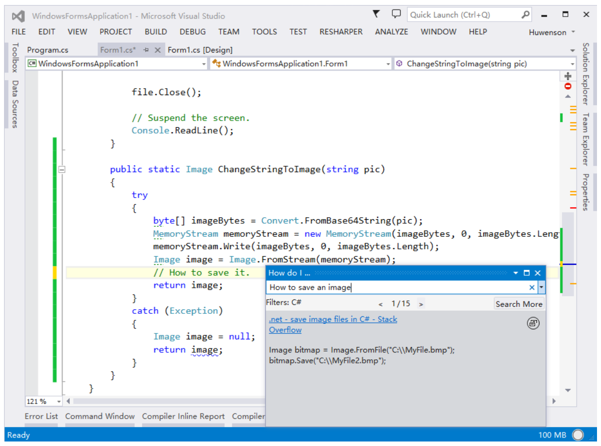
Figure 5 shows a screenshot of BDA for recommending API sample code. When a user types an incomplete API, Visual Studio (through its IntelliSense feature) will recommend a list of candidate APIs. If the user does not know how to use these APIs, she can examine the sample code returned by BDA within the Visual Studio environment. In Figure 5, the user enters an incomplete API '*doc.l…*', BDA automatically recommends a code snippet for a candidate function 'Load'.

Currently, BDA provides sample code/projects collected from a variety of websites through Bing search. These websites include Stack Overflow, dotnetperls, C#411, Cppreference, cplusplus.com, and MSDN. Table 2 shows the total number of code snippets that can be obtained from two popular software forums (Stack Overflow and MSDN), as of June 2016.

Table 2. The total number of code snippets in two software forums (June 2016)

Domain	C#	C/C++	JavaScript
MSDN	63,417	22,016	2,832
Stack Overflow	452,736	243,462	880,896

Figure 6 shows a screenshot of BDA for answering user's free text code search query. The user enters the query *how to save an image* through the *How do I* interface. The request is sent to the BDA backend and related code snippets extracted from web pages (in this case, a code snippet from Stack Overflow) are returned to the user.


WindowsFormsApplication1 - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST RESHARPER ANALYZE WINDOW HELP
Huwenson
Program.cs Form1.cs* Form1.cs [Design]
WindowsFormsApplication1 WindowsFormsApplication1.Form1 ChangeStringToImage(string pic)
How do I ...
Filters: C#
1/15
Search More
121 %
Error List Command Window Compiler Inline Report Compiler
Ready 100 MB

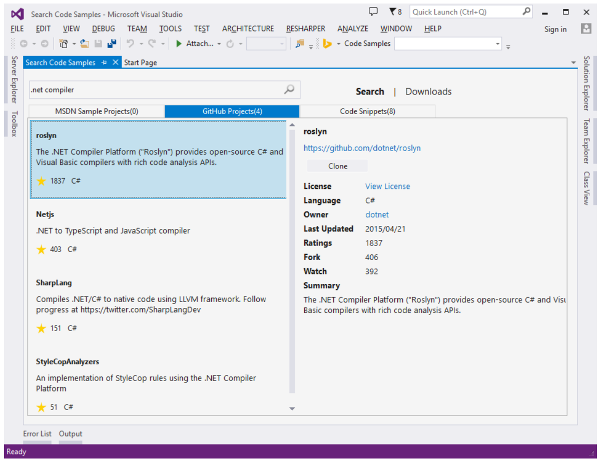
```
            file.Close();

            // Suspend the screen.
            Console.ReadLine();
        }

        public static Image ChangeStringToImage(string pic)
        {
            try
            {
                byte[] imageBytes = Convert.FromBase64String(pic);
                MemoryStream memoryStream = new MemoryStream(imageBytes, 0, imageBytes.Leng
                memoryStream.Write(imageBytes, 0, imageBytes.Length);
                Image image = Image.FromStream(memoryStream);
                // How to save it.
                return image;
            }
            catch (Exception)
            {
                Image image = null;
                return image;
            }
        }
    }
```

How to save an image

.net - save image files in C# - Stack Overflow

```
Image bitmap = Image.FromFile("C:\\MyFile.bmp");
bitmap.Save("C:\\MyFile2.bmp");
```

Figure 6: Recommending sample code based on user query


Search Code Samples - Microsoft Visual Studio
FILE EDIT VIEW DEBUG TEAM TOOLS TEST ARCHITECTURE RESHARPER ANALYZE WINDOW HELP
Sign in
Attach...
Code Samples
Search Code Samples Start Page
Search | Downloads
MSDN Sample Projects(0) GitHub Projects(4) Code Snippets(8)
Clone
Error List Output
Ready

.net compiler

roslyn
The .NET Compiler Platform ("Roslyn") provides open-source C# and Visual Basic compilers with rich code analysis APIs.
★ 1837 C#

Netjs
.NET to TypeScript and JavaScript compiler
★ 403 C#

SharpLang
Compiles .NET/C# to native code using LLVM framework. Follow progress at https://twitter.com/SharpLangDev
★ 151 C#

StyleCopAnalyzers
An implementation of StyleCop rules using the .NET Compiler Platform
★ 51 C#

roslyn
https://github.com/dotnet/roslyn

License	View License
Language	C#
Owner	dotnet
Last Updated	2015/04/21
Ratings	1837
Fork	406
Watch	392

Summary
The .NET Compiler Platform ("Roslyn") provides open-source C# and Visu Basic compilers with rich code analysis APIs.

Figure 7: Recommending sample projects based on user query

Figure 7 shows a screenshot of recommending sample projects. Through the search box of BDA, users can issue a free-text query

[7] https://msdn.microsoft.com/en-us/library/hcw1s69b.aspx

(in this case, "*.net compiler*") and access a large number of popular open source projects sites such as GitHub and MSDN. Figure 7 shows that BDA retrieves projects such as Roslyn (the .NET Compiler Platform). To facilitate reuse, BDA can also pull these libraries into the user's Visual Studio environment.

4.2 User Feedback

Bing Developer Assistant was first released in December 2014[8]. The GitHub integration was released on April 2015 during the Microsoft Build 2015 conference[9]. BDA has been well received by developers. It has received 670K+ downloads since its initial release. Now it handles about 3 million sample code requests per month. In Visual Studio Gallery, the overall user rating for BDA is 4 (out of 5)[10].

We also received many encouraging comments from the survey. For example:

"It's awesome add-on, really helpful and speeding up development, whenever you have doubts about the way to write the code you can look it up directly from VS…" – szalap

"Really neat! Love the way it inserts the code sample straight into my code." - Carl Clark

"Awesome, streamlined tool that makes it super easy to download sample projects. No more zip files! Yes!" – ShrikeSoft

The users also gave us many constructive comments. For example:

"Very nice and useful tool. I'm looking forward to support for more languages." - kevin-mcc

"Still getting used to it but I can see the potential. Any guidelines as to how to get a code sample to show up would be handy." - JayChase

"Very resource-intensive assistant. Very nice idea though. Hopefully performance will improve over time." -- Ilia101

"When can we get this for python & R?" – Shahrokh Mortazavi

These comments suggest that the diversity, usability, and performance of the tool could be further improved. We will address these constructive comments and improve the tool in our future work.

Overall, the results of real-world usage support our belief that BDA can help improve developer productivity by providing a seamless integration of sample code mining and Visual Studio.

5. RELATED WORK

In software development practice, developers frequently search for sample code for reuse. Many code search tools, such as Ohloh [6] and Krugle [5], have been proposed to help developers find relevant code. Portfolio [8] visualizes relevant functions and their usages using a combination of models that address surfing behavior of developers. CodeHow [7] performs free-text based code search over GitHub projects. It considers the impact of both text similarity and potential APIs on code search. BDA supports code search through Bing search engine. It also supports the recommendation of API sample code mined from GitHub and web pages.

There is a lot of research on APIs. For example, API popularity [19], API evolution [20], and API usage pattern mining [15, 16]. ParseWeb [11] accepts queries of the form "Source -> Destination" from a developer and gives the code samples containing the given Source and Destination object types. MAPO [16] and UP-Miner [15] mine the patterns of API method calls from source code. Buse and Weimer [2] presented an automatic technique for mining and synthesizing API usage examples. Gu et al. [22] applied deep learning to generate API usage sequences for a given natural language query. Recently, to facilitate research on APIs, Sawant and Bacchelli [12] developed a dataset for API usage by mining GitHub projects.

Some researchers also explored the use of web search engines to help with programming tasks [11]. For example, Mica [13] is a web application that uses Google API to find relevant pages and then analyze the content of those pages to extract the most relevant programming terms. Assisme [4] and Blueprint [1] use general search engines to search for results and automatically extract relevant code snippets from the returned results. eXoaDocs [18] facilitates API reuse by embedding API documents with code examples mined from the Web. Unlike these related work, BDA seamlessly integrates web search results into a programming IDE (Visual Studio). BDA also supports the extraction of sample code from open source projects. Seahawk [21] is an Eclipse plug-in that integrates stackoverflow.com support into the IDE. It can recommend relevant Stack Overflow posts within an IDE, while BDA recommends sample code.

6. CONCLUSIONS

In this paper, we describe Bing Developer Assistant (BDA), a tool that helps improve developer productivity by recommending API sample code mined from public software repositories (such as GitHub) and web pages (such as Stack Overflow). BDA provides code snippets that implement an API or answer a code search query. Using BDA, developers can find and reuse a large amount of sample code from within the Visual Studio IDE.

We are now extending BDA as a service and integrating it with other Microsoft services such as Bing.com. Through a Bing search, developers can obtain the API sample code provided by BDA service for a variety of programming languages. The returned sample code is displayed within web browser. The API usage data and sample code could also be used by other software engineering research projects. The vision of Bing Developer Assistant is to provide developer help across technologies and canvases. Currently it only supports Visual Studio. We would soon extend the support to other IDE ecosystems.

7. ACKNOWLEDGEMENT

We thank all Microsoft developers who participated in the development and maintenance of Bing Developer Assistant, especially Yi Wei, Vinod Patankar, Sumit Saluja, Sachin Joseph, Shabbar Husain, Mei Liang, and Dongmei Zhang. We also thank our intern students who helped with the implementation and maintenance of BDA, especially Fei Lv, Wenhao Song, Sheng Tian,

[8] http://blogs.msdn.com/b/visualstudio/archive/2014/12/16/bing-developer-assistant-for-visual-studio.aspx

[9] http://blogs.msdn.com/b/visualstudio/archive/2015/04/30/github-integration-in-developer-assistant.aspx

[10] https://visualstudiogallery.msdn.microsoft.com/a1166718-a2d9-4a48-a5fd-504ff4ad1b65

Peiyong Lin, Senlan Yao, Qing Ren, Wangsheng Hu, Xutong Chen, Chengxun Shu, Hong Wu, Xingzhao Yue, and Chen Xia.

8. REFERENCES

[1] J. Brandt, M. Dontcheva, M. Weskamp, and S. R. Klemmer, “Example centric programming: Integrating web search into the development environment,” in *Proc. CHI ’10*, 2010, pp. 513–522.

[2] P.L. Buse and W. Weimer. “Synthesizing API Usage Examples,” In *Proc. ICSE’12*, pp 782-792, 2012.

[3] R.Holmes and G. C. Murphy. “Using structural context to recommend source code examples,” In *Proc. ICSE*, 2005, pp. 117-125.

[4] R. Hoffmann, J. Fogarty, and D. S. Weld, “Assieme: Finding and leveraging implicit references in a web search interface for programmers,” in *Proc. 20th Annual ACM Symposium on User Interface Software and Technology (UIST ’07)*, 2007, pp. 13–22.

[5] Krugle code search. [Online]. Available: http://www.krugle.com/

[6] Ohloh code search. [Online]. Available: https://code.ohloh.net/

[7] F. Lv, H. Zhang, J. Lou, S. Wang, D. Zhang, and J. Zhao, "CodeHow: Effective Code Search based on API Understanding and Extended Boolean Model", in *Proc. ASE 2015*, Lincoln, Nebraska, Nov 2015, pp. 260-270.

[8] C.McMillan, M. Grechanik, D. Poshyvanyk, Q. Xie, and C. Fu, “Portfolio: finding relevant functions and their usages,” In *Proc. ICSE 2011*, pp. 111-120.

[9] M.P. Robillard, “What makes APIs hard to learn? Answers from developers,” *IEEE Software,* vol. 26, no. 6, pp. 27–34, 2009.

[10] M.P. Robillard and R. DeLine. A Field Study of API Learning Obstacles. Empirical Soft. Engin.*, 16(6):* 703-732, 2011.

[11] S. E. Sim and R. Gallardo-Valencia, *Finding Source Code on the Web for Remix and Reuse*, Springer, 2013.

[12] A. Sawant and A. Bacchelli, A Dataset for API Usage, in *Proc. 12th Working Conference on Mining Software Repositories (MSR)*, 2015, pp. 506-509.

[13] J. Stylos and B. A. Myers, “Mica: A web-search tool for finding API components and examples,” in *Proceedings of the Visual Languages and Human-Centric Computing (VLHCC ’06)*, 2006, pp. 195–202.

[14] S.Thummalapenta, T. Xie. “PARSEWeb: a programmer assistant for reusing open source code on the web,” In *Proc. ASE 2007*, pp. 204-213.

[15] J. Wang, Y. Dang, H. Zhang, K. Chen, T. Xie, and D. Zhang. Mining succinct and high-coverage API usage patterns from source code. *In Proc. of the 10th Working Conference on Mining Software Repositories*, pp. 319-328, 2013.

[16] H. Zhong, T. Xie, L. Zhang, J. Pei, H. Mei. “MAPO: mining and recommending API usage patterns,” In *Proc. ECOOP 2009*, pp. 318-343.

[17] Y. Wei and N. Chandrasekaran and S. Gulwani and Y. Hamadi, Building Bing Developer Assistant, Microsoft technical report MSR-TR-2015-36, May 2015.

[18] J. Kim, S. Lee, S.-w. Hwang, and S. Kim. Towards an intelligent code search engine. In *Prof. 24th AAAI Conference on Artificial Intelligence*, 2010.

[19] Y. M. Mileva, V. Dallmeier, and A. Zeller, “Mining API popularity,” in *Testing–Practice and Research Techniques*. Springer, 2010, pp. 173–180.

[20] D. Dig and R. Johnson, “How do APIs evolve? a story of refactoring,” *Journal of software maintenance and evolution: Research and Practice*, vol. 18, no. 2, pp. 83–107, 2006.

[21] L. Ponzanelli, A. Bacchelli, and M. Lanza, “Seahawk: Stack overflow in the IDE,” in *Proceedings of ICSE 2013 (35th International Conference on Software Engineering*, Tool Demo Track), pp. 1295–1298, 2013.

[22] X. Gu, H. Zhang, D. Zhang, S. Kim. Deep API Learning, in *Proc. 24th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2016)*, Seattle, WA, USA, November 2016.