The Journal of Systems and Software 162 (2020) 110505

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

SCC++: Predicting the programming language of questions and snippets of Stack Overflow

Kamel Alrashedy*, Dhanush Dharmaretnam, Daniel M. German, Venkatesh Srinivasan, T. Aaron Gulliver

University of Victoria Victoria, BC, Canada V8W 2Y2, Canada

ARTICLE INFO

Article history:
Received 23 March 2019
Revised 11 November 2019
Accepted 19 December 2019
Available online 23 December 2019

Keywords:
Classification
Machine learning
Natural language processing
And programming languages

ABSTRACT

Stack Overflow is the most popular Q&A website among software developers. As a platform for knowledge sharing and acquisition, the questions posted on Stack Overflow usually contain a code snippet. Determining the programming language of a source code file has been considered in the research community; it has been shown that Machine Learning (ML) and Natural Language Processing (NLP) algorithms can be effective in identifying the programming language of source code files. However, determining the programming language of a code snippet or a few lines of source code is still a challenging task. Online forums such as Stack Overflow and code repositories such as GitHub contain a large number of code snippets. In this paper, we design and evaluate Source Code Classification (SCC++), a classifier that can identify the programming language of a question posted on Stack Overflow. The classifier achieves an accuracy of 88.9% in classifying programming languages by combining features from the title, body and the code snippets of the question. We also propose a classifier that only uses the title and body of the question and has an accuracy of 78.9%. Finally, we propose a classifier of code snippets only that achieves an accuracy of 78.1%. These results show that deploying Machine Learning techniques on the combination of text and code snippets of a question provides the best performance. In addition, the classifier can distinguish between code snippets from a family of programming languages such as C, C++ and C#, and can also identify the programming language version such as C# 3.0, C# 4.0 and C# 5.0.

1. Introduction

In the last decade, Stack Overflow has become a widely used resource in software development. Today inexperienced programmers rely on Stack Overflow to address questions they have regarding their software development activities.

Forums like Stack Overflow and Code Review rely on the tags of questions to match them to users who can provide answers. However, new users in Stack Overflow or novice developers may not tag their posts correctly. This leads to posts being downvoted and flagged by moderators even though the question may be relevant and adds value to the community. In some cases, Stack Overflow questions that are related to programming languages may lack a programming language tag. For example, Pandas is a popular Python library that provides data structures and powerful data analysis tools; however, its Stack Overflow questions usually do not include a Python tag. This could create confusion among developers who may be new to a programming language and might not be familiar with all of its popular libraries. The problem of missing language tags could be addressed if posts are automatically tagged with their associated programming languages.

Code snippet tools, such as gist and pastebin, allow users to organize and share their code snippets with other users. These tools cannot predict the programming languages of these snippets and assume that the code snippets have already been tagged with the correct programming language by the user.

Existing solutions to this prediction problem are not satisfactory. Integrated Development Environment (IDE) such as CLion, Eclipse, and text editors such as Notepad++, SublimeText and Atom predict the language based on the file extension rather than the source code itself. This can cause inconvenience to the users as they need to create the file with the correct extension manually to enable syntax highlighting in these editors.

* Corresponding author.
E-mail addresses: kamel@uvic.ca (K. Alrashedy), dhanushd@uvic.ca (D. Dharmaretnam), dmg@uvic.ca (D.M. German), srinivas@uvic.ca (V. Srinivasan), agullive@uvic.ca (T. Aaron Gulliver).

https://doi.org/10.1016/j.jss.2019.110505
0164-1212/

Machine Learning (ML) and Natural Language Processing (NLP) based techniques have been widely applied in the study of source codes (cf. Hindle et al., 2009; Nguyen et al., 2013; Bielik et al., 2016; Oda et al., 2015; Nguyen and Nguyen, 2015). It has been shown that these techniques can help with a variety of important tasks involving programming languages such as understanding the summary of source code, code completion and suggestion, code engine search and classifying programming languages.

Classifying programming languages of source code files using ML and NLP methods has been well explored in the research community (cf. Kennedy et al., 2016; Gilda, 2017; Khasnabish et al., 2014). It has been established that the programming language of a source code file can be identified with high accuracy. However, most of the previous work that study the classification of programming languages use the GitHub dataset in which the size of source code files is typically large. Applying ML and NLP methods to classify a large source code file provides a very high accuracy as the large sample contains many features that help the machine learning model to learn better. In this paper, we are interested in a tool that can classify a code snippet which is a small block of reusable code with at least two lines of code, a much more challenging task. The only previous work that studies classification of the programming languages from a code snippet or a few lines of source code is the work of Baquero et al. (2017). However, they achieve low accuracy showing that identifying programming languages from a small source code or a code snippet is much harder than from larger pieces.

The only known tool that can predict the programming language of a code snippet is Programming Languages Identification (PLI), available in Algorithmia, a marketplace for AI based algorithms (Programming language identification tool, 2018). PLI supports 21 programming languages: Bash, C, C#, C++, CSS, Haskell, HTML, Java, JavaScript, Lua, Objective-C, Perl, PHP, Python, R, Ruby, Scala, SQL, Swift, VB.Net and Markdown. It is claimed that PLI can predict 21 languages with a reported accuracy of 99.4% top1 accuracy on GitHub source code. However, code snippets from Stack Overflow are much smaller in size compared to GitHub source code and the accuracy of PLI for code snippet classification has not been looked at.

In this paper, we conduct an empirical study of a classifier, SCC++, to predict the programming languages in Stack Overflow questions. Our research questions are:

1. **RQ1.** Can we predict the programming language of a question in Stack Overflow?
2. **RQ2.** Can we predict the programming language of a question in Stack Overflow without using code snippets inside it?
3. **RQ3.** Can we predict the programming language of code snippets in Stack Overflow questions?

For the first research question, we are interested in evaluating how machine learning performs while trying to identity the language of a question when all the information in a Stack Overflow question is used; this includes its title, body (together referred to as textual information) and the code snippets in it. The purpose of the second research question is to determine if the inclusion of code snippets is essential to determining the programming language that a question refers to. Finally, the purpose of the third research question is to evaluate the ability to use machine learning to predict the language of a snippet of source code; a successful predictor will have applications beyond Stack Overflow, as it could also be applied to snippet management tools and code search engines that scan documentation and blogs for relevant information.

The main contributions of this paper are as follows:

1. A prediction method that uses a combination of code snippets and textual information in a Stack Overflow question. This classifier achieves an accuracy of 88.9%, a precision of 0.89 and a recall of 0.89 in predicting the programming language tag.
2. A classifier that uses only textual information in Stack Overflow questions to predict their programming language. This classifier achieves an accuracy of 78.9%, a precision of 0.81 and a recall of 0.79 which is much higher than the previous best model (Baquero et al., 2017).
3. A prediction model based on Random Forest (Breiman, 2001) and XGBoost (Chen and Guestrin, 2016) classifiers that predicts the programming language using only a code snippet in a Stack Overflow question. This model is shown to provide an accuracy of 78.1%, a precision of 0.79 and a recall of 0.78.
4. Comparison of SCC++ to Programming Languages Identification (PLI) to show that SCC++ achieves much higher accuracy than PLI. PLI can only achieve an accuracy of 55.5%, a precision of 0.61 and a recall of 0.55 in classifying 21 programming languages.
5. Evaluation results demonstrating that SCC++ can also distinguish between code snippets from the family of programming languages, C, C# and C++ with an accuracy of 80%, and can identify the programming language version, C# 3.0, C# 4.0 and C# 5.0 with an accuracy of 61%.

The rest of the paper is organized as follows. We begin by discussing the related work in Section 2. In Section 3, we describe dataset extraction and processing as well as machine learning classifiers. Our methodologies and results are presented in Section 4. Sections 5 and 6 present the discussion and future work. Finally, threats to validity and conclusions are outlined in the last two sections of the paper (Sections 7 and 8).

2. Related work

Predicting a programming language of a given source code file has been a rising topic of interest in the research community.

Kennedy et al. (2016) proposed a model to identify the software languages of entire source code files from Github using natural language identification techniques. Their classifier, based on five statistical language models from NLP trained on a GitHub dataset, identified 19 programming languages with an accuracy of 97.5%. Gilda (2017) used a dataset from GitHub repositories for training a convolutional neural network classifier. Their classifier could classify 60 programming languages of source code files from Github with 97% accuracy.

Khasnabish et al. (2014), collected more than 20,000 source code files to train and test their model. These source codes were extracted from multiple repositories in GitHub. The model was trained and tested using a Bayesian classifier model and was able to predict 10 programming languages with 93.48% accuracy.

Some editors such as Sublime and Atom add highlights to code based on the programming language. However, this requires an explicit extension, e.g..html,.css,.py. Portfolio (McMillan et al., 2011) is a search engine that supports programmers in finding functions that implement high-level requirements in query terms. This engine does not identify the language, but it analyzes code snippets and extracts functions which can be reused. Holmes et al. (2005) developed a tool called Strathcona that can find similar snippets of code.

David et al. (2011) collected 41,000 source code files from GitHub for the training dataset and 25 source code files are randomly selected for the testing dataset. However, their classifier, that used supervised learning and intelligent statistical feature selection, only achieved 48% accuracy.

In (Baquero et al., 2017), the authors predicted the programming language of code snippets of Stack Overflow posts. 1000 question posts were extracted for each of 18 different programming languages. They trained their classifier using a Support Vector Machine algorithm. Their model achieved a very low accuracy of 44.6% compared to the previous works because predicting the programming language of a code snippet is more complex and challenging than predicting the programming language of a source code file.

Rekha et al. (2014) proposed a hybrid auto-tagging system that suggests tags to users who create questions. When the post contains a code snippet, the system detects the programming language based on the code snippets and suggests many tags to users. Multinomial Naive Bayes (MNB) was trained and tested for the proposed classifier which achieved 72% accuracy. Saha et al. (2013) converted Stack Overflow questions into vectors, and then trained a Support Vector Machine using these vectors and suggested tags used the model obtained. The tag prediction accuracy of this model is 68.47%. Although it works well for some specific tags, it is not effective with some popular tags such as Java. Stanley and Byrne (2013) used a cognitive-inspired Bayesian probabilistic model to choose the most suitable tag for a post. This is the tag with the highest probability of being correct given the a priori tag probabilities. However, this model normalizes the top for all questions, so it is unable to differentiate between a post where the top predicted tag is certain, and a post where the top predicted tag is questionable. As a consequence, its accuracy is only 65%.

3. Tool description

SCC++ is a classification tool for identifying the programming language of code snippets and textual information. It is currently able to identify 21 different programming languages. SCC++ is an open source tool and therefore, it is possible to train it on a new dataset to support and identify a new programming language. SCC++ is trained using a dataset curated from Stack Overflow and is implemented using Scikit-Learn Pedregosa et al. (2011), a machine learning library in Python.

Fig. 1 shows how SCC++ functions. SCC++ is described in detail in the following subsections.

3.1. Dataset extraction and processing

In this section, the details of the Stack Overflow dataset are discussed. Then, the steps used for data extraction and processing are explained.

3.1.1. Stack overflow selection

As of July 2017, Stack Overflow had 37.21 million posts, of which 14.45 million are questions with 50.9k different tags. In this paper, the programming language tags in Stack Overflow are of interest. The 21 programming languages that were selected in our study are very popular in Stack Overflow according to 2017 Stack Overflow developer survey (Developer, 2017). They constitute around 80% of the questions in Stack Overflow. The languages selected for our study are: Bash, C, C#, C++, CSS, Haskell, HTML, Java, JavaScript, Lua, Objective-C, Perl, PHP, Python, R, Ruby, Scala, SQL, Swift, VB.Net, Markdown.

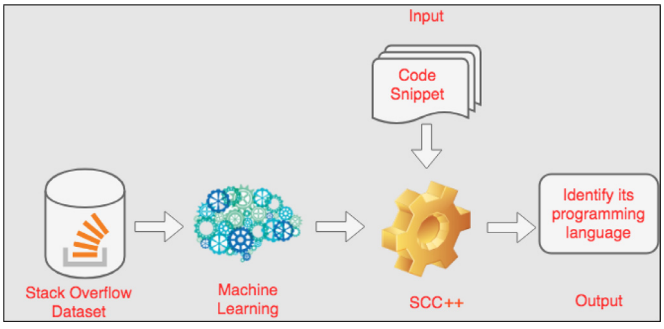


Fig. 1. Functioning of SCC++.

3.1.2. Extraction and processing of stack overflow questions

The Stack Overflow July 2017 data dump was used for analysis. In our study, questions with more than one programming language tag were removed to avoid potential problems during training. Questions chosen contained at least one code snippet, and the code snippet had at least 2 lines of code. For each programming language, 12,000 random questions were extracted; however, two programming languages had less than 12,000 questions: Markdown (1,210) and Lua (8,107). In total, 232,727 questions were selected for the study. Fig. 3(a) shows an example of a Stack Overflow post[1]. It contains (1) the title of the post (2) the text body (3) the code snippet and (4) the tags of the post. It should be noted that the tags of the question were removed and not included as a part of text features during the training process to eliminate any inherent bias.

The.xml data was parsed using xmltodict and the Python Beautiful Soup library to extract the code snippets and text from each question separately. See Fig. 4. A Stack Overflow question consists of a title, body and code snippet. The tags <code> and <code> were utilized to extract the code snippet from Stack Overflow question. In some cases, a question contained multiple code snippets; these were combined into one. The questions were used to create three datasets: textual information, code snippets and combination of textual information and code snippets.

The title and body (which we refer to as textual information) and code snippets were used to answer the first research question. The textual information was used to answer the second research question. Finally, the code snippets were used to answer the last research question.

Machine learning models cannot be trained on raw text because their performance is affected by noise present in the data. The textual information (title and body) needed to be cleaned and prepared before the machine learning model can be trained to provide a better prediction. A few preprocessing steps were required to clean the text. First, non-alphanumeric characters such as punctuation, numbers and symbols were removed. Second, entity names were identified using the dependency parsing of the Spacy Library (Honnibal and Johnson, 2015). An entity name is a proper noun (for example, the name of an organization, company, library, function etc.). Third, stop words such as *after, about, all, and from etc.* were removed. Fourth, since the entity name can have different forms (such as study, studies, studied and studious), it is useful to train using one of these words and predict the text containing any of the word forms. To achieve this goal, stemming and lemmatization was performed using the NLTK library in Python (Loper and Bird, 2004). At the end of all the preprocessing steps, the remaining words were used as features to help train the machine learning model. Fig. 3(a) is the original Stack Overflow post and Fig. 3(b) is the Stack Overflow post after application of NLP techniques. However, code snippets were treated as a natural language constructs, which means that punctuation was removed. The features of a code snippet are the keywords, identifier, name of the library etc. It should be noted that stemming and lemmatization were not applied on code snippets.

[1] https://stackoverflow.com/questions/1642697/.

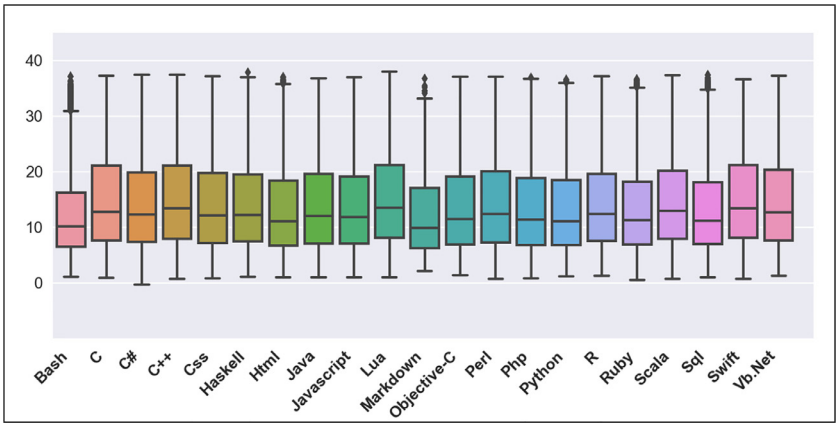
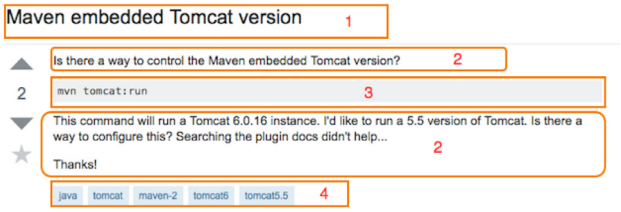


Fig. 2. The length of code snippets in the SO dataset.



(a) Before applying NLP techniques.

maven tomcat version way maven tomcat version command tomcat instance version tomcat way plugin docs

(b) After applying NLP techniques.

Fig. 3. An example of a Stack Overflow Question.

The extracted set of questions provide a good coverage of different versions of programming languages. For example, code snippets were extracted for the python tags: Python-3.x, Python-2.7, Python-3.5, Python-2.x, Python-3.6, Python-3.3 and Python-2.6, for the Java tags: Java-8 and Java-7, and for the C++ tags: C++11, C++03, C++98 and C++14. The snippets extracted had a significant variation in number of lines of code, as shown in Fig. 2.

3.2. Machine learning classifiers

The ML algorithms Random Forest Classifier (RFC) and XGBoost (a gradient boosting algorithm) were employed. These algorithms provided higher accuracy compared to the other algorithms we explored, ExtraTree and MultiNomialNB. The performance metrics used in this paper are precision, recall, accuracy, F1 score and confusion matrix (for the third research question).

3.2.1. Random forest classifier (RFC)

RFC (Breiman, 2001) is an ensemble algorithm which combines more than one classifier. This classifier generates a number of decision trees from randomly selected subsets of the training dataset. Each subset provides a decision tree that votes to make the final decision during test. The final decision made depends on the decision of a majority of trees. One advantage of this classifier is that if one or few of the trees make a wrong decision, it will not affect the accuracy of the result significantly. Also, it avoids the overfitting problem seen in the Decision Tree model. The total number of trees in the forest is an important parameter because a large number of trees in the forest give high accuracy.

3.2.2. XGBoost

XGBoost (Chen and Guestrin, 2016), standing for "Extreme Gradient Boosting", is a tree based model similar to Decision Tree and RFC. The idea behind boosting is to modify the weak learner to be a better learner. Recall that Random Forest is a simple ensemble algorithm that generates many subtrees and each tree predicts the output independently. The final output will be decided by the majority of the votes from the subtrees. However, XGBoost is a better model because each subtree makes the prediction sequentially. Hence, each subtree learns from the mistakes that were made by the previous subtree. The idea of XGBoost came from gradient boosting, but XGBoost uses the regularized model to help control overfitting and give a better performance.

3.3. Usage example

SCC++ is a simple command-line tool, and was built based on Stack Overflow dataset and machine learning classifiers. To run SCC++, the first step is to load the dataset and select the feature set. The next step is to train the machine learning algorithm on the selected features. Subsequently, users will be asked to enter their code snippet through command line. Finally, the predicted programming language is output. SCC++ is open source[2] and the

[2] https://github.com/Kamel773/SourceCodeClassification-2019.

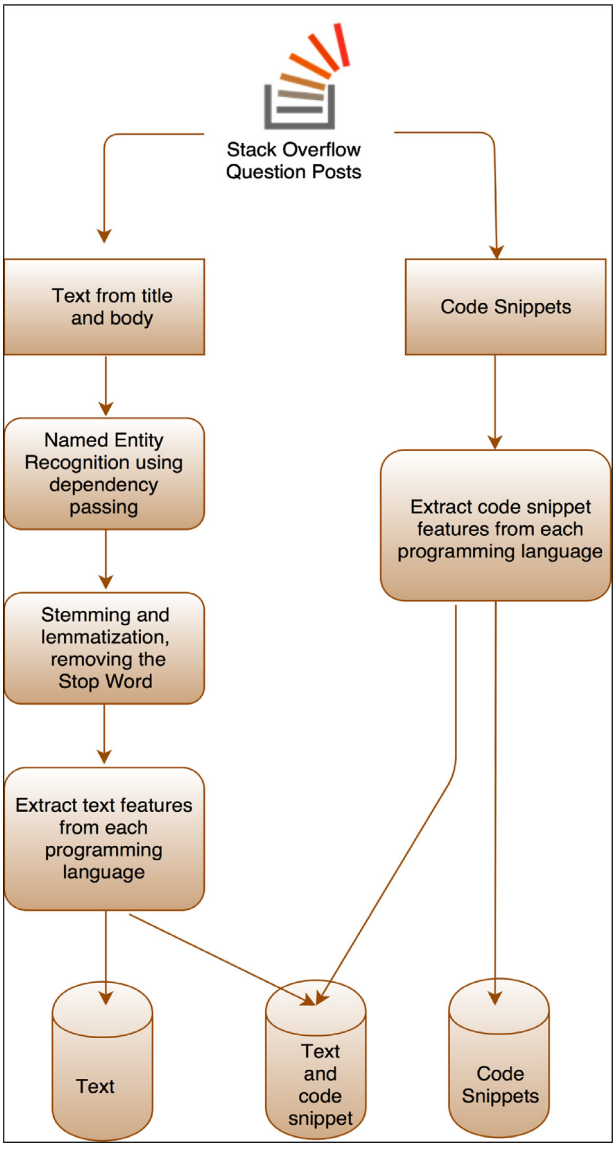


Fig. 4. The dataset extraction process.

dataset used is available online.[3] SCC++ can be used in three different modes; code snippets only, textual information only or a combination of code snippet and textual information.

4. The empirical study

In this section, the results obtained for the three research questions are described in detail and in addition, SCC++ is compared to other existing tools.

4.1. Evaluation of SCC++

SCC++ was evaluated to answer the three research questions, RQ1-3. For the first research question, our goal was to evaluate if SCC++ can classify the language of a question when all the information in a Stack Overflow question is used; this includes its title, body (textual information) and code snippets in it, across 21 programming languages. The purpose of the second research question was to evaluate if the inclusion of code snippets is an essential factor in determining the programming language that a question refers to. Finally, in the third research question, the task of classifying the programming language of code snippets of Stack Overflow was explored. In addition, it was checked if SCC++ can identify different versions of one programming language, C# 3.0, C# 4.0 and C# 5.0 and also if SCC++ can distinguish between code snippets from a family of programming languages such as C, C++ and C#.

4.1.1. Methodology

Textual information (title and body) and code snippets extracted from the Stack Overflow questions were split using the Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer from the Scikit-learn library (Pedregosa et al., 2011). The Minimum Document Frequency (Min-DF) was set to 10, which means that only words present in at least ten documents were selected (a document can be either the code snippet, the textual information, or both—code snippet and textual information). This step eliminates infrequent words from the dataset which helps machine learning models learn from the most important vocabulary. The Maximum Document Frequency (Max-DF) was set to default because the stop words were already removed in the data preprocessing step discussed in Section 3.1.

The machine learning models were tuned using RandomSearchCV, which is a tool for parameter search in the Scikit-learn library. The XGBoost algorithm has many parameters, such as minimum child weight, max depth, L1 and L2 regularization, and evaluation metrics such as Receiver Operating Characteristic (ROC), accuracy and F1 score. RFC is a bagging classifier and has a parameter number of estimators which is the number of subtrees used to fit the model. It is important to tune the models by varying these parameters. However, parameter tuning is computationally expensive using a technique such as grid search. Therefore, a deep learning technique called Random Search (RS) tuning was used to train the models. All model parameters were fixed after RS tuning on the cross-validation sets (stratified ten-fold cross-validation). For this purpose, the dataset was split into training, validation and test partitions. Test data consisted of 20% of total data. The training dataset was split into k =10 folds. During training, 9 folds were used for training and the left out fold was used for validation. The trained model was evaluated on cross validation fold and discarded. This process is then repeated till we train and evaluate on all folds of data. This technique is called *k*-fold validation. We used $k = 10$ and all reported cross valuation accuracy are average values (James et al., 2013).

As a part of the third question, there were two sub research questions. The purpose of the first sub research question was to evaluate if SCC++ is able to distinguish between a code snippets across one family of programming languages, C, C# and C++. To answer this research question, we created a subset of our dataset which only contains the three programming languages C, C# and C++. Then, the dataset was split into training (including validation data) and test data in the ratio of 80:20 as described above. The other research question was to evaluate if SCC++ can identify different versions of one programming language, C# 3.0, C# 4.0 and C# 5.0. A new dataset was extracted from Stack Overflow using three question tags, C# 3.0, C# 4.0 and C# 5.0, to answer this sub research question.

4.1.2. Results

RQ1. Can we predict the programming language of a question in Stack Overflow?

To answer this question, XGBoost and RFC classifiers were trained on the combination of textual information and code snippet datasets. The RFC classifier achieves an accuracy of 87.7%, and the average score for precision, recall and F1 score were 0.88, 0.88 and 0.88 respectively; on the other hand, the XGBoost achieved an

[3] https://drive.google.com/open?id=1leMs0rdKAfX1UYEhSEe9a1npIWrIvqr6.

Table 1
The F1 score for all the three research questions. The Min and Max of the scores for each column are highlighted.

Lang. / Algos	XGBoost			Random Forest		
	RQ3	RQ2	RQ1	RQ3	RQ2	RQ1
Bash	0.80	0.90	0.95	0.85	0.64	0.90
C	0.76	0.85	0.86	0.81	0.85	0.90
C#	0.80	0.81	0.91	0.78	0.65	0.84
C+	**0.49**	0.90	0.91	0.73	**0.50**	0.79
CSS	0.87	0.91	0.97	0.77	0.82	0.83
Haskell	**0.90**	0.87	0.97	0.78	0.90	0.96
HTML	0.57	0.64	**0.69**	0.55	0.63	**0.63**
Java	0.70	**0.53**	0.78	0.76	0.62	0.81
Javascript	0.80	0.88	0.94	0.74	0.81	0.89
Lua	0.86	0.91	0.97	0.70	0.76	0.79
Markdown	0.77	0.61	0.83	**0.91**	0.88	0.97
Objective-C	0.65	0.83	0.91	0.88	**0.95**	**0.98**
Perl	0.77	0.85	0.92	**0.41**	0.88	0.78
PHP	0.75	0.64	0.82	0.88	0.79	0.94
Python	0.89	**0.92**	**0.98**	0.79	0.88	0.94
R	0.78	0.85	0.89	0.78	0.83	0.91
Ruby	0.72	0.83	0.89	0.72	0.82	0.87
Scala	0.78	0.85	0.92	0.81	0.80	0.90
SQL	0.67	0.76	0.80	0.79	0.83	0.91
Swift	0.87	0.81	0.94	0.89	0.91	0.96
VB.Net	0.86	0.67	0.92	0.77	0.82	0.86

accuracy of 88.9%, and the average score for precision, recall and F1 score were 0.89, 0.89 and 0.89 respectively. The results for XGBoost classifier are discussed in further detail because it provides the best performance. Most programming languages had a high F1 score: Python (0.98), Lua (0.97), CSS (0.97) and Haskell (0.97) had the highest, while HTML (0.69), Java (0.78), SQL (0.80) and PHP (0.82) had the lowest. Table 1 gives the F1 score for each programming language using XGBoost and Random Forest classifiers on the combination of textual information and code snippets.[4]

RQ2. Can we predict the programming language of a question in Stack Overflow without using code snippets inside it?

To answer this research question, two machine learning models were trained using the XGBoost and RFC classifiers on the dataset that contained only the textual information. The XGBoost classifier achieved an accuracy of 78.9%, and the average score for precision, recall and F1 score were 0.81, 0.79 and 0.79 respectively. RFC achieved a slightly lower performance than XGboost, with an accuracy of 78.2% and an average score for precision, recall and F1 score of 0.79, 0.78 and 0.78 respectively. Note that the accuracy of XGBoost using textual information decreased by about 10% compared to using both the textual information and its code snippets. Table 1 shows the F1 score for each programming language using XGBoost and Random Forest classifiers on textual information.

The top performing languages based on the F1 score were Python (0.92), Lua (0.91), CSS (0.91), C++ (0.90), Bash (0.90), JavaScript (0.88) and Haskell (0.87). It should be further noted that the languages Python, Haskell, CSS and Lua had a high F1 score and performed very well in both the (RQ1) and (RQ2).

RQ3. Can we predict the programming language of code snippets in Stack Overflow questions?

To predict the programming language from a given code snippet, two ML classifiers were trained on the code snippet dataset. XGBoost achieved an accuracy of 77.4%, and the average score for precision, recall and F1 score were 0.78, 0.77 and 0.78 respectively. RFC achieved an accuracy of 78.1%, and the average score for precision, recall and F1 score were 0.79, 0.78 and 0.78 respectively. Table 1 provides the F1 score of each programming language while using XGBoost and Random Forest on code snippets.

When a Random Forest classifier was used, the programming languages Markdown (0.91), Objective-C (0.88) and PHP (0.88) had a good F1 score. The F1 score of PHP in (RQ1) and (RQ3) were very high; however, in (RQ2) the F1 score dropped down by 10% which means the features from code snippets helps machine learning algorithms. In XGBoost, Objective-C had the worst F1 score and precision (0.65 and 0.58); but its recall was high (0.75). When the programming language of a code snippet was extremely hard to identity, XGBoost frequently misclassified it as Objective-C, while we observed that RFC misclassified such snippets as Haskell. We looked manually at some of these code snippets and were not able to identify the programming language. For example, such a code snippet only contains a declaration and assignment of a variable or an array or a body of a for loop. This is the main motivation for combining the textual information and code snippet in (RQ1). If the classifier gets confused when predicting the programming languages from a code snippet, the textual information (title and body) will help the machine learning model to make a better prediction. Figs. 5 and 6 show the confusion matrix for the XGBoost and RFC classifiers. Table 2 shows how the accuracy improves as the minimum size of the code snippet in the dataset is increased from 2 to 20 lines.

The comparison between the results of textual information dataset and code snippet dataset shows a similar accuracy (in average). On the other hand, using the combination of textual information and code snippet significantly increased the accuracy by 10% compared to using only the textual information. Since many Stack Overflow posts can have a large textual information and a small code snippet or vice versa, combining the two gives a high accuracy in (RQ1).

RQ3-1. Can SCC++ distinguish between code snippets across one family of programming languages such as C, C# and C++?

This experiment involved training and testing SCC++ on three programming languages from the same family, C, C++ and C#, Multinomial Naive Bayes (MNB) classifier was used to answer this question. In this experiment SCC++ achieved a high accuracy of 80.0% and the average scores for precision, recall and F1 score were 0.81, 0.80 and 0.80 respectively. Table 3 shows the details of the performance on C, C# and C++. C# has unique features compared to C and C++ and hence SCC++ can classify C# with a high F1 score of 0.88. The percentage of C++ code snippets misclassified as C was 20%.

RQ3-2. Can SCC++ identify different versions of a programming language, specifically C#?

For this experiment, SCC++ was trained and tested using Multinomial Naive Bayes (MNB) classifier on a dataset that only contained the three versions of C#. SCC++ achieved an accuracy of 61.0% and the average of scores for precision, recall and F1 score were 0.61, 0.61 and 0.61 respectively. The details of the performance is shown in Table 3. We noticed that SCC++ found it particularly hard to distinguish between the versions, C# 4.0 and C# 5.0. These results show that, while SCC++ is highly accurate in distinguishing between code snippets from C, C# and C++ family, it is less accurate at identifying the different versions of C# 3.0, C# 4.0 and C# 5.0.

4.2. Comparison to PLI

4.2.1. Methodology

As mentioned in the introduction, it is claimed that PLI provides a high accuracy while predicting the programming language of a given source code file. However, predicting the language of a code snippet is far more challenging. We evaluated the performance of PLI in predicting the programming languages of code snippets and compared its results with SCC++. 150 code snippets were randomly selected from each programming language. This created a subset of

[4] We focus on F1 scores in this section. All of our results for precision and recall can be found on our GitHub repository.

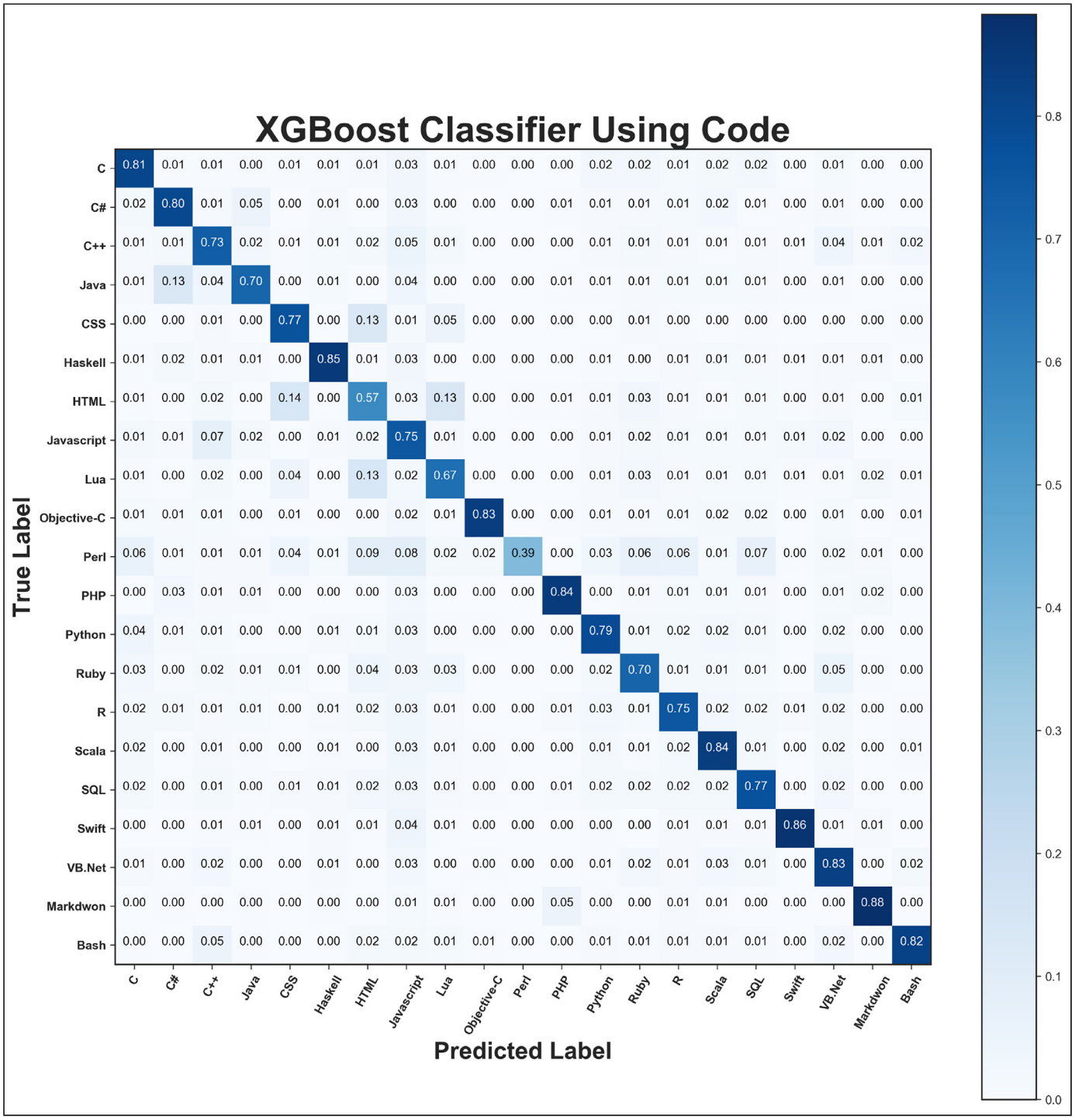


Fig. 5. Confusion matrix for the XGBoost classifier trained on code snippet features. The diagonal represents the percentage of snippets of a programming language correctly predicted.

4200 code snippets which could be used for prediction using PLI. We used only 150 snippets because we had to pay for the classification of each one of them. We used the urlib library in Python to make the API calls to PLI to generate predictions for all 4200 code snippets. The API call returned a JSON file with languages as key and corresponding probability for all 21 languages that it supported. The programming language with highest probability score was selected as the predicted language. We could compare SCC++ with PLI with respect to the first research question because both tools support the 21 programming languages. However, we could not study the performance of PLI with respect to research questions RQ3-1 and RQ3-2 because this tool is closed-source and we were not able to train it using a dataset that contained only a family of programming language or different versions of a programming language.

4.2.2. Results

PLI achieved an accuracy of 55.5% and the average scores for precision, recall and F1 score were 0.61, 0.55 and 0.55 respectively.

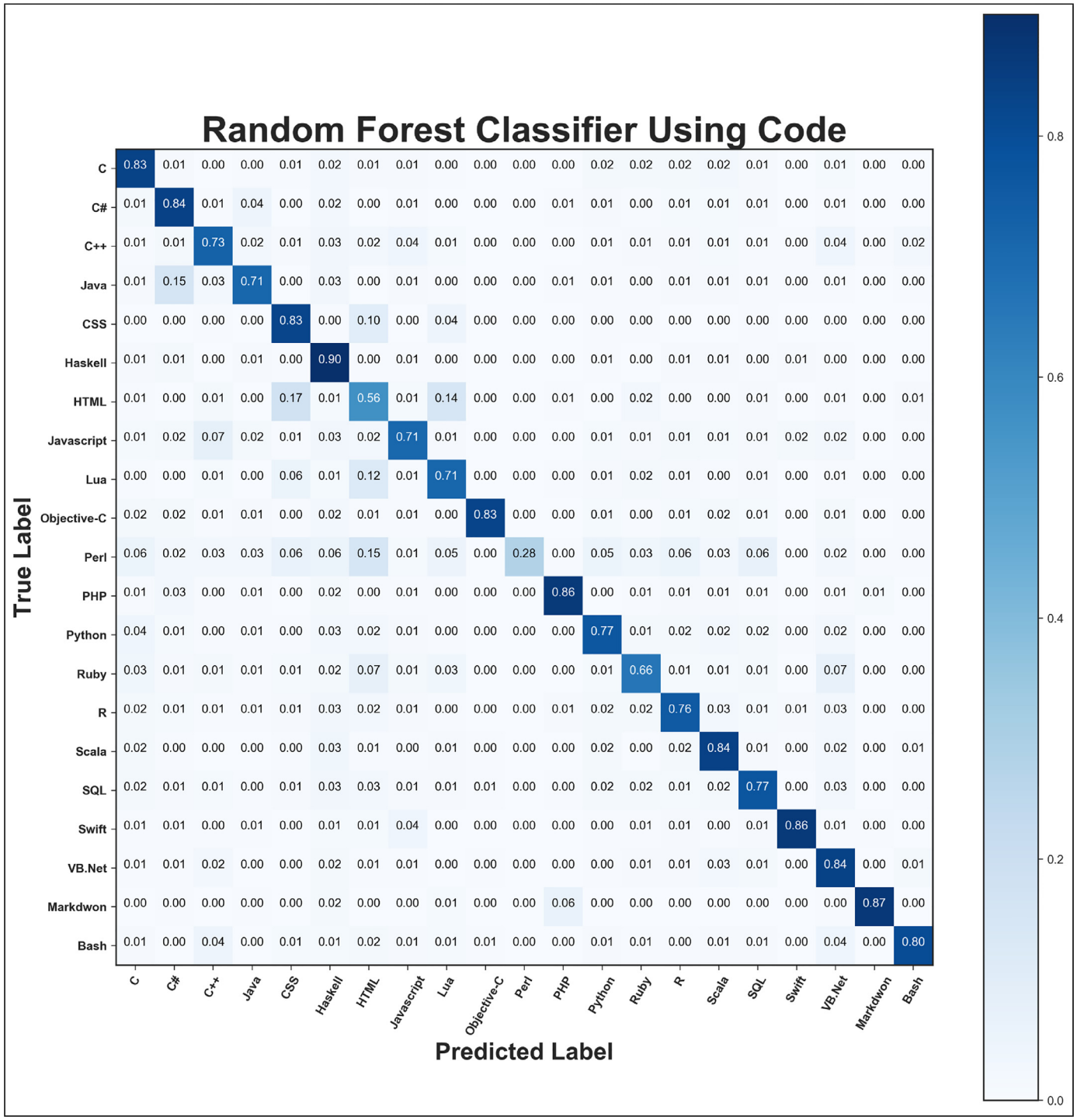


Fig. 6. Confusion matrix for the RFC classifier trained on code snippet features. The diagonal represents the percentage of snippets of a programming language correctly predicted.

The F1 score of PLI for each programming language is shown in Table 4.

CSS had the worst recall score of 0.19 among the programming languages. This is because 40% of code snippets of CSS were misclassified as HTML. The syntax and operations of these two programming languages are extremely similar to each other. Similarly; 21% of the code snippets of JavaScript were misclassified as HTML. We also noticed that PLI misclassified many code snippets from CSS and HTML as Javascript, pointing to its inherent weakness.

Objective-C had the highest F1 score of 0.77. This language has very unique syntax compared to other programming languages in our study. Some programming languages that were correctly classified with high precision were VB.Net (0.89), R (0.88), Objective-C (0.85) and Bash (0.79). The overall F1 score for SCC++ was higher than PLI and SCC[5]; however, Perl had a good F1 score of 0.69 and

[5] SCC, a preliminary version of SCC++, is a tool that classifies the programming language of a code snippet from Stack Overflow using the MNB algorithm (Alrashedy et al., 2018).

Table 2
Effect of the minimum number of lines in a code snippet on accuracy.

Minimum Lines	Accuracy	Precision	Recall	F1 score
More than 2	78.1%	0.79	0.78	0.78
More than 5	79.9%	0.80	0.80	0.80
More than 10	82.6%	0.83	0.83	0.83
More than 15	85.1%	0.85	0.85	0.85
More than 20	86.0%	0.86	0.86	0.86

Table 3
The performance of SCC++ when used for identifying different versions of a programming language.

Language	F1 score	Language	F1 score
C	0.78	**C#-3.0**	0.77
C#	0.88	**C#-4.0**	0.56
C+	0.75	**C#-5.0**	0.58

Table 4
The comparison of F1 score for SCC++, SCC (Alrashedy et al., 2018) and PLI.The Min and Max of the scores for each column are highlighted.

Lang. / Tools	PLI	SCC	SCC+
Bash	0.67	0.76	0.85
C	0.56	0.76	0.81
C#	0.51	0.79	0.78
C+	0.65	**0.51**	0.73
CSS	0.30	0.86	0.77
Haskell	0.67	**0.89**	0.78
HTML	0.35	0.54	0.55
Java	0.46	0.70	0.76
JavaScript	0.48	0.78	0.74
Lua	0.50	0.84	0.70
Markdown	**0.28**	0.76	**0.91**
Objective-C	**0.77**	0.57	0.88
Perl	0.69	0.74	**0.41**
PHP	0.62	0.74	0.88
Python	0.69	0.88	0.79
R	0.72	0.77	0.78
Ruby	0.43	0.70	0.72
Scala	0.72	0.76	0.81
SQL	0.50	0.65	0.79
Swift	0.54	0.84	0.89
Vb.Net	0.60	0.83	0.77

0.74 in PLI and SCC respectively, and had the lowest F1 score of 0.41 among all the programming languages in SCC++.

19% of code snippets from C were classified as C++, and 7% of code snippets of C++ were classified as C. Ruby, HTML, CSS and Markdown had the worst F1 scores of 0.43, 0.35, 0.30 and 0.28 respectively. Since HTML and CSS share a similar syntax and PLI was inept in classifying these languages, the F1 score for these languages dropped down.

5. Discussion

The most important observation in the previous section is that for the research question (RQ1), XGBoost achieves a high accuracy of 88.9%, while for (RQ2) and (RQ3), it only achieves an accuracy of 78.9% and 77.4% respectively. This observations highlights the importance of using the combination of textual information and code snippets in predicting tags in comparison to textual information or code snippet only. In some cases, Stack Overflow posts contain very small code snippets making it extremely hard to identify its language as many programming languages share the same syntax.

Dependency parsing and extracting of entity names using a Neural Network (NN) through Spacy appeared to help reduce noise and to extract important features from Stack Overflow questions. This is likely the main reason for the significant improvement in performance compared to previous approaches in the literature.

The analysis of the feature space of the top performing languages indicates that these languages have unique code snippet features (keywords/identifiers) and textual information features (libraries, functions). For example, when the textual information based features were visualized for Haskell, words such as 'GHC', 'GHCI', 'Yesod' and 'Monad' were obtained. 'GHC' and 'GHCI' are compilers for Haskell, 'Yesod' is a web-based framework and 'Monad' is a functional programming paradigm (Haskell is a functional programming language). Most of the top performing languages have a small feature space (vocabulary) as compared to more popular languages such as Java, Vb.Net and C# which have a large number of libraries and standard functions, and support multiple programming paradigms resulting in a large feature space. A large feature space adds more complexity to the ML models. We have compiled the top-50 code snippet features for all the programming languages and uploaded them to our GitHub repository.[6]

Recall that TF-IDF based approach was applied on the dataset before machine learning algorithms were trained, and achieved very high accuracy on both textual information and code snippets. However, another approach based on Word2Vec was also applied, but achieved much lower accuracy. Improving the accuracy of a Word2Vec based classification is an interesting open problem.

The current version of SCC++ is only covers the most popular 21 programming languages in Stack Overflow. In order to cover more languages, we studied an extension of our tool by adding a new class called "Others" during training and testing. This class contained a collection of code snippets from four other languages, which are Assembly, Go, Coffee Script and Matlab. Our experiment showed a drop in accuracy of roughly 4% for the three questions. The tool achieved an accuracy of 84.1% for RQ1, and an accuracy of 75.3% and 74.0% for RQ2 and RQ3 respectively. In summary, our tool can be extended to cover all languages with some loss in accuracy.

5.1. Limitations of SCC++

SCC++ was trained to predict a programming language of Stack Overflow question and its snippets. In some cases, Stack Overflow questions may contain multiple code snippets from different languages. Such questions will have more than one language tag. It is natural to ask if it is possible to extend SCC++ to predict multiple language tags. The problem of multi-label classification is much more challenging and is not considered in the current version of SCC++.

6. Future work

The study of programming language prediction from textual information and code snippets is still new, and much remains to be done. Most of the existing tools focus on file extensions rather than the code itself. In recent years, there has been tremendous progress made in the field of deep learning, especially for time series or sequence-based models such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. RNN and LSTM models can be trained using source code one character at a time as input, but they can have a high computational cost.

NLP and ML techniques perform much better in predicting languages compared to tools that predict directly from code snippets. Stack Overflow text is somewhat unique in the sense that it captures the tone, sentiments and vocabulary of the developer community. This vocabulary varies depending on the programming language. Therefore, it is important that the vocabulary for each programming language is captured, understood and separated.

[6] https://github.com/Kamel773/SourceCodeClassification-2019.

The approach of this paper is to treat the code snippet as containing natural language; the punctuation and syntax of the programming language are not considered. A future direction of research should consider punctuation and syntax as features to identify the programming language of code snippets.

In the future, our model will be evaluated using programming blog posts, library documentation and bug repositories. This would help us understand how general the model is.

7. Threats to validity

Construct Validity: In creating the datasets from Stack Overflow, only the most popular programming languages were extracted, and this was based solely on the programming language tag. However, some tags synonymous with languages were not included in the extraction process. For example, 'SQL SERVER', 'PLSQL' and 'MICROSOFT SQL SERVER' are related to 'SQL' but were discarded. Furthermore, we cannot be absolutely confident that all code snippets of Stack Overflow are correct, these snippets may contain incorrect syntax or could be debugging messages.

Furthermore, we only chose Stack Overflow posts containing code snippets with least two lines of code. This is because we found some code snippets with one line of code, and it was difficult, using human intelligence, to identify its language. For example, a code snippet with one line of code only contained an assignment of number to a variable, such as "$i = 5$" and "print(i)", and this could be hard to classify.

Internal validity: We studied how SCC++ and PLI are able to classify the programming language of a code snippet. While we studied if SCC++ can distinguish between snippets from a family of programming languages (C, C# and C++) and also examined if SCC++ can identify the three versions of programming language C#, we were unable to run those experiments for PLI as it is closed-source. Also, we were only able to evaluate PLI with 150 snippets because we had to pay for every classified snippet.

After the datasets were extracted, dependency parsing[7] was used to select entity names so as to include only the most important features from textual information. The use of dependency parsing can result in the loss of critical vocabulary and might affect our results. However, we manually analyzed the vocabulary before and after the dependency parsing to ensure that information related to the languages was not lost.

External validity: We only used Stack Overflow as the source of data for our analysis. We did not explore other sources such as GitHub repositories or other sources of code such as extracting a snippet from source code file of GitHub. Therefore, we cannot be absolutely confident that our results would be the same across all sources of code snippets. Also note that our comparison is only against PLI since this is the only tool available at this time. This is mainly due to the lack of open source tools for predicting languages. The current version of SCC++ is only able to classify the most popular 21 programming languages and does not cover all the programming languages found in Stack Overflow posts.

8. Conclusions

In this paper we discussed the importance of predicting languages from code snippets and textual information. In particular, we chose to focus on predicting the programming language of Stack Overflow questions. We argued that predicting the programming language from code snippets is far more challenging than from source code files considering complexity of modern programming languages. We proposed Source Code Classification (SCC++), a tool built using a Random Forest and XGBoost classifiers trained on a Stack Overflow dataset.

Our results show that training and testing the classifier by combining the textual information and code snippet achieves the highest accuracy of 88.9%. Other experiments using either textual information or code snippets achieve accuracies of 78.9% and 78.1% respectively. This implies that information from textual features is easier for a machine learning model to learn compared to information from code snippet features. Our results show that it is possible to identify the programming language of a snippet of few lines of source code. Finally. We compared SCC++ against PLI, the only known proprietary tool for this problem and found that SCC++ achieved a much higher accuracy than PLI (that achieved only an accuracy of 55.5%) on code snippets from Stack Overflow posts. We believe that our tool could be applied in other scenarios such as code search engines and snippet management tool.

[7] Dependency parsing is the task of recognizing a sentence and assigning a syntactic structure to it (Jurafsky and Martin, 2009).

Acknowledgments

Kamel Alrashedy acknowledges the financial support of the Saudi Ministry of Education through a graduate scholarship. The authors would like to thank Kwang Moo Yi for his helpful comments on this work.

References

Alrashedy, K., Dharmaretnam, D., German, D.M., Srinivasan, V., Gulliver, T., 2018. SCC: automatic classification of code snippets. In: Proceedings of the 18th International Working Conference on Source Code Analysis and Manipulation (SCAM), pp. 203–208.

Baquero, J.F., Camargo, J.E., Restrepo-Calle, F., Aponte, J.H., Gonzalez, F.A., 2017. Predicting the programming language: extracting knowledge from stack overflow posts. In: Proceedings of Colombian Conference on Computing (CCC), pp. 199–221.

Bielik, P., Raychev, V., Vechev, M.T., 2016. PHOG: probabilistic model for code. In: Proceedings of the 33nd International Conference on Machine Learning, ICML, pp. 2933–2942.

Breiman, L., 2001. Random forests. In: Machine Learning, pp. 5–32.

Chen, T., Guestrin, C., 2016. XGBoost: a scalable tree boosting system. In: Proceedings of 22nd SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 785–794.

David, K., Kyle, M., Simon, W., 2011. Algorithmic Programming Language Identification. CoRR http://arxiv.org/abs/1106.4064.

Developer, 2017. Survey results. [online]., In Available: https://insights.stackoverflow.com/survey/2017#technology.

Gilda, S., 2017. Source code classification using neural networks. In: Proceedings of International Joint Conference on Computer Science and Software Engineering (JCSSE), Evolution, and Reengineering, pp. 1–6.

Hindle, A.J., Godfrey, M.W., Holt, R.C., 2009. What's hot and what's not: windowing developer topic analysis. In: Proceedings of the 25th IEEE International Conference on Software Maintanence (ICSM), pp. 339–348.

Holmes, R., Walker, R.J., Murphy, G.C., 2005. Strathcona example recommendation tool. In: ACM SIGSOFT Software Engineering Notes, pp. 237–240.

James, G., Daniela, W., Trevor, H., Robert, T., 2013. An Introduction to Statistical Learning. Springer, New York. 181

Jurafsky, D., Martin, J.H., 2009. Speech and Language Processing. Pearson Prentice Hall. 223

Kennedy, J., Dam, V., Zaytsev, V., 2016. Software language identification with natural language classifiers. In: Proceedings of IEEE International Conference on Software Analysis, Evolution, and Reengineering, 19, pp. 624–628.

Khasnabish, J.N., Sodhi, M., Deshmukh, J., Srinivasaraghavan, G., 2014. Detecting programming language from source code using Bayesian learning techniques. In: Proceedings of International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM), pp. 513–522.

Honnibal, M., Johnson, M., 2015. An improved non-monotonic transition system for dependency parsing. In: Proceedings of Conference on Empirical Methods in Natural Language Processing, pp. 1373–1378.

Loper, E., Bird, S., 2004. NLTK: the natural language toolkit. In: Proceedings of 42nd Annual Meeting of the Association for Computational Linguistics, pp. 63–70.

McMillan, C., Grechanik, M., Poshyvanyk, D., Xie, Q., Fu, C., 2011. Portfolio: a search engine for finding functions and their usages. In: Proceedings of 33rd International Conference on Software Engineering, pp. 1043–1045.

Nguyen, A.T., Nguyen, T.N., 2015. Graph-based statistical language model for code. In: International Conference on Software Engineering, pp. 858–868.

Nguyen, T.T., Nguyen, A.T., Nguyen, H.A., Nguyen, T.N., 2013. A statistical semantic language model for source code. In: Proceedings of the 9th Joint Meeting on Foundations of Software Engineering, pp. 532–542.

Oda, Y., Fudaba, H., Neubig, G., Hata, H., Sakti, S., Toda, T., Nakamura, S., 2015. Learning to generate pseudo-code from source code using statistical machine translation. In: Automated Software Engineering (ASE), pp. 574–584.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: machine learning in python. J. Mach. Learn. Res. 2825–2830.

Programming language identification tool, 2018. Available: https://www.algorithmia.com [Online].

Rekha, V.S., Divya, N., Bagavathi, P.S., 2014. A hybrid auto-tagging system for StackOverflow forum questions. In: Conference on Interdisciplinary Advances in Applied Computing. 56:1–56:5

Saha, A.K., Saha, R.K., Schneider, K.A., 2013. A discriminative model approach for suggesting tags automatically for stack overflow questions. In: Proceedings of the 10th Working Conference on Mining Software Repositories, pp. 73–76.

Stanley, C., Byrne, M.D., 2013. Predicting tags for StackOverflow posts. In: Proceedings of International Conference on Cognitive Modeling, pp. 414–419.

Kamel Alrashedy is a master's student in the Department of Computer Science at the Universityof Victoria, where he does research in the areas of Applied Machine Learning, Data Mining and mining software repositories.

Dhanush Dharmaretnam is a master's student in the Department of Computer Science at theUniversity of Victoria, where he does research in the areas of natural language processing and deep learning.

Daniel M. German is Professor in the Department of Computer Science at the University of Victoria, where he does research in the areas of mining software repositories, open source software ecosystems and the impact of intellectual property in software engineering.

Venkatesh Srinivasan is Professor in the Department of Computer Science at the University of Victoria, where he does research in the areas of algorithms for massive data sets, social network mining, data privacy and lower bounds in concrete models for big data.

Aaron Gulliver is Professor in the Department of Electrical & Computer Engineering at the University of Victoria, where he does research in the areas of algebraic coding theory, information theory, cryptography and machine learning.