Understanding and Predicting the Programming Languages in Stack Overflow

Dhanush Dharmaretnamr[1] and Kamel Alrashedy[1]
[1]Department of Computer Science
University of Victoria
PO Box 1700, STN CSC, Victoria BC, Canada V8W 2Y2
Dhanushd@uvic.ca, Kamel@uvic.ca

***Abstract*— In this paper, we evaluated existing tools and methods available to predict programming languages by analyzing textual data and code snippets extracted from Stack Overflow posts. We evaluated a popular tool called Algorithmia which predicts programming languages from source code and found that performance of the tool was well below it advertised accuracy of 99.4%. We also trained machine learning models on textual data and achieved state of the art accuracy in predicting programming language. We compared the confusion matrix and other metrics generated by both the methods and found that they do not agree with one another. This result paves way for use of meta classifiers which uses both text and code snippets to make accurate predictions on programming languages. We also found that the tools and methods doesn't generalize in predicting from code snippets across different versions/Standards of the same programming language. The tool also fails to make accurate decisions on smaller snippets of code. We also found that there exists a strong positive correlation between lines of code and prediction capability of the tool.**

***Index Terms*— Stack Overflow, Machine learning, Tag, Code Snippet, Programming Languages and Natural Language Processing, Algorithmia.**

I. INTRODUCTION

In the early days of software programming before the dawn of Google, it was difficult for the programmers to debug and implement complex pieces of software. Projects often relied on experienced programmers, code documentation and textbooks to learn and understand not just the features and functionalities of a language but the programming language itself. In the last decade, the advent of Social media forums for programmers such as Stack Overflow, Quora, Hackers News and various others have made learning and coding much simpler and faster. Young inexperienced developer no longer have to wait for experts in their projects to help them debug their code or to implement some esoteric functionalities/libraries in programming languages in their daily coding routine.

A. Why do we need to predict the programming languages from source code or text?

Along with the growth of online social forums for programming, both the complexity and the features supported by various programming languages have also increased over the last decade due to easy access to documentation, growth of open source repositories such as Git Hub which enabled easy access of source code of new libraries. There have been also an increased adaptation of esoteric languages such as Haskell and Scala, genesis of new languages based on existing languages such as Typescript (from JavaScript),Hack (from Php), Julia (from Matlab, C and Fortan) etc. and also splitting of languages based on standards such as C++11,C++14, C++17 standards for C++, Java7, Java8, Java-EE6, Java9 standards for Java.

This increased complexity has a wider impact in the world of programming. For example forums like Stack Overflow rely on the tags of questions to match questions to other users who might provide an answer for it. There are also users in Stack Overflow who watches certain tags because they are moderators for those tags or want to earn additional reputation by answering questions in their areas of expertise. There are new users in Stack Overflow or novice developers who might not tag the posts correctly. This leads to posts being down voted and flagged by moderators even though the question may be unique and might add more value to the community. This problem could be solved if we have better tag predicting capability especially for programming languages. There are IDE's such as Clion, Eclipse etc. and text editors such as notepad++, SublimeText, Atom etc. which highlights the syntax of code by identifying the programming language being typed on. It may be worth noting that almost all the IDE's and text editors predict the language based on its file extension rather than the source code itself. This cause inconvenience to the users as they need to create the file with correct extension manually to enable syntax highlighting in these editors.

There are also many tools available such as Linguist by Github, ProgrammingLanguageIdentification tool by Algorithmia(an AI market) which predicts programming languages based on code snippets [12]. The Algorithmia tools currently predicts 21 languages with a reported accuracy of 99.4% Top1 accuracy on Github code snippets. Most of these tools are not trained on a standard dataset and cannot accommodate complexities of the programming languages that we have described above. In this paper we explore and evaluate ProgrammingLanguageIdentification tool by Algo-

rithmia using code snippets from Stack Overflow questions. We would study posts of varying complexities in terms of lines of code, number of characters per code snippet and various standards belonging to programming languages. This also gives us the opportunity to study and understand similarity and differences between various programming languages that are popular today.

RQ 1. ***Can we predict the tags of programming languages using code embedded in the Stack Overflow post using existing tools?***

It is a general belief that the complexity of code increases with the increase in the lines of code. What does this mean for the tools which predict languages from source code? Does the increase or decrease in the lines of code have any impact in their prediction capability? This paper also explores the impact of lines of code on the prediction accuracy of tools such as ProgrammingLanguageIdentification by Algorithmia. In most cases, these tools are trained on complete source code files or library files such as one from GitHub. This makes the tools perform better on large source code files and libraries as compared to small snippets of code that we often see in Stack Overflow and other software forums. However, it is also important for us to understand the generalisability of these tools across all possible sources of code.

RQ 2. ***How does the lines of code impact the prediction capability of tools which predict programming languages from code snippets?***

Another draw back of the tools predicting programming languages is that they don't generalize well across various standards of the same languages. This could be again attributed to the fact that the training data for the tools does not incorporate files or code snippets from all the standards of a programming language. For example tools trained on C++ may not work for predicting C++11 or C++14 code snippets. Therefore, it becomes important to understand the existing capabilities of tools in identifying programming languages across various standards. We believe that highlighting vulnerabilities of the tools would enable more research into the development of tools that predict programming languages.

RQ 3. ***How does the tools that predict programming languages from code snippets perform across various standards of the same language?***

So far we have talked about predicting languages from source code. There are a lot of textual data available about programming languages, its libraries and functionalities through websites like GitHub, Jira, BugZilla, Stack Overflow, Quora etc. The study of these documents would enable us to understand the programming languages, its origin and complexities. For example mining the documentation in GitHub repositories would enable us to study the evolution of programming languages or adaptation of features and functionalities of languages. The Stack Overflow posts are an other source of information about programming languages. The post description texts could be mined to add a new dimension of understanding on programming languages and also could enable us to predict the languages without using the source code. The last five years have also shown a tremendous growth in the field of Machine Learning, Deep learning and Natural Language processing (NLP)[13]. This has resulted in an improved performance in tasks such as document classification, Machine translation and other predictive tasks. Therefore, we could use Machine learning and NLP techniques to predict the programming from textual data.

RQ 4. ***Can we predict the tags of programming languages by analyzing the texts describing a stack Overflow question?***

B. Why did we select Stack Overflow as ground truth dataset for our proposed study ?

Stack Overflow is a question and answer online forum for developer community. Developers posts question on various coding, implementation and debugging aspects of programming and these questions are answered by experts in various domain. According to Quantcast[17]., a popular web traffic analytics website, Stack Overflow had 2.9 Billion Global Visits and 7.3 Billion Global views between the period of Dec 15 2016 to Dec 14 2017. Stack Overflow is the most common forum among the developers with 1.2 million unique visits to the page between Nov 23 2016 to Dec 14 2017. The number of new users have steadily grown in Stack Overflow since its inception in 2008 (Fig1).

As of July 2017 Stack Overflow consisted of 37.21 million posts of which 14.45 million are question posts and We identified 50906 Unique tags in Stack Overflow. However, we identified that 150 tags constituted about 87% of the stack Overflow Posts. This is quite evident by looking at Fig2. Therefore, we can conclude that most of the SO tags are not that popular or are relatively new. However, we are mostly interested in the programming language tags in the Stack Overflow. We have selected 20 languages for our analysis and they constitutes about 52% of questions in Stack Overflow. Even though rest of 48% tags are also related to some programming language but they are not given a language tag. For example Pandas is a popular library in Python for reading large CSV files but questions associated with Pandas mostly does not have the Python tags in them. This could create confusion among developers who may be new to a programming language and are not familiar with all its popular libraries. The problem of missing language tag could be addressed if posts are automatically tagged with their associated programming language. The language we have selected for our study are listed below:

Bash, C, C#, C++, CSS, Haskell, HTML, Java, JavaScript, Lua, Objective-c, Perl, PHP, Python, R, Ruby, Scala, SQL, Swift, Vb.net.

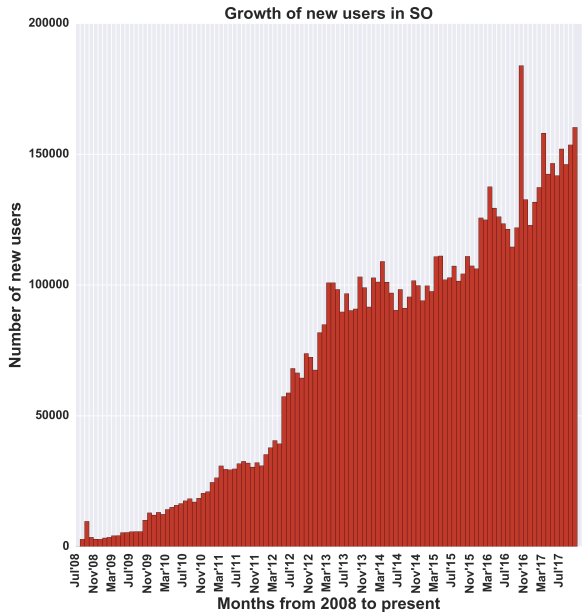


Fig. 1: Plot showing the growth of users in Stack Overflow. The number of users signing up has increased linearly over the last one decade.

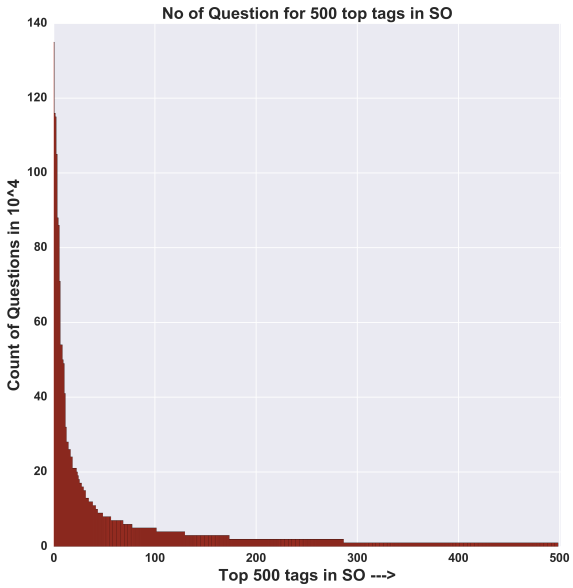


Fig. 2: Plot showing number of questions per tag in Stack Overflow for the top 500 tags. Top 150 tags constitutes about 87% of all question posted in Stack Overflow.

The selection of these languages are mainly influenced by the availability of the tools that could be used to predict them from their source code. ProgrammingLanguageIdentification tool from Algorithmia supports all of the above mentioned languages. Moreover, they include both new and old generation languages and has varying degree of popularity.

C. Why did we select the tool from Algorithmia market place for our analysis and experiments ?

Algorithmia is a popular online market place which enable use of machine learning models as a micro service using REST API. ProgrammingLanguageIdentification is one among many tools hosted by Algorithmia and currently supports 21 programming languages. The approximate cost for making 10,000 API calls is 20 USD. The tool was originally trained using source code from various GitHub repositories and claims 99.4% accuracy. We searched for other tools which could be used for predicting programming languages but we couldn't find others which could support as many as 20 programming languages. We also didn't prefer those tools which predicts languages based on their file extension as it does not suit the goals of this paper. Therefore, we selected Algorithmia as the only tool for predicting language from code snippets in our research

II. Background and Related Work

Predicting a programming language from a given text and from code snippets has been a rising topic of interest in the research community. Despite of a lot of interesting contributions to this field, we are still lacking behind in predicting the exact programming language just on the basis of text or the code snippet. Let's take a rising text editor(Sublime) for instance. It provides some interesting highlights to code on the basis of programming language. However, we need to explicitly mention the extension(eg .html, .css., .py and so on). None of the previous works build a model to predicting the tag by training and testing the snippet of code to identify the programming languages. To our best Knowledge, we are the first researchers predicting the tag using a snippet of code in the SO post. Also, Natural Language Processing approach are used to predicting the tag by training and testing posts title and body. Then comparing both approaches to understand what is the best choose to build a model and why.

On similar line, Portfolio[6] is a search engine that supports programmers in finding functions that implement high level requirements in query terms. This search engine does not identify the coding languages, however it does deal with analyzing code snippet and extracting the valuable functions which can be reused. Another tool developed by Holmes *et al.* [8] called Strathcona could find similar snippet of code.

Stack OverFlow questions and answers have a score feature which is calculated as difference between up-vote and down-vote. The majority of software developers use Stack Overflow as the main source to find a solution to their problem [14]. The post also could have an accepted answer as selected by the person who posts the question. By using these as features, we could study what makes a good question and answer posts. In 2013, Asaduzzaman*et al* found around 7.5% of Stack Overflow questions are unanswered. Also, they build a prediction model that can predict the average time to get

the first answer. It's important therefore to understand what factors differentiates a good post from a bad one. Nachi et al. [3]. studied what makes a post good or bad based on lines of code, quality of code and texts in a Stack Overflow post. Similarly, C. Treude *et al.* [5] analyzed Stack OverFlow questions to explore what kinds of question could receive a good answer. They found that the posts which contained snippets of code constituted a good answer.

Software maintainers prefer to talk to each other to understand the code [9]. Also, the researchers found three reasons why the programmers use Q&A web site; just-in-time learning, clarify their knowledges and remembering the details. Programmers generally use the web as the main source to gather information and find a solution for their problem[4]. They us both code, documentation and links to ask their questions in online forums. This gives us opportunity to use both code and text to study and learn programming languages and also to predict them from these data sources. Rekha et al [10] authors proposed a Hybrid auto-system tag system, this system suggests a tag for the user after entering the question. Saha [1] *et al.* followed the approach of converting Stack Overflow Questions into vectors, and then training Support Vector Machines on the basis of those vectors and eventually suggesting tags on the basis of built model. The accuracy for tag prediction for there model ranged from 50% to 100%. Although there tagging system works pretty well for some specific tags, however it does not work well with some basic tags, like Java. Another suggestion by Stanley and Byrne [2] is to use a cognitively-inspired Bayesian probabilistic model. The approach is to choose one most suitable tag for each post. The paper suggests to choose the tag with highest log odds of correctness, when each tag's prior probability is known. However, the model used normalizes the top activations for all posts, because of which the model cannot differentiate between a post where the top predicted tag is certain, and a post where the top-predicted tag is questionable. This model is 65% accurate when tasked to predict one tag per post on average. Most of the prior work falls short in creating a robust system which could assign tags to programming languages.

III. Dataset extraction and processing

In this section we discuss in detail about how we extracted the data from Stack Overflow dump, processed the data and answered our research questions. We used the Stack overflow July 2017 data dump for all our analysis. The Stack Exchange Data dump for Stack OverFlow was downloaded in .xml from archive.org. We selected 5000 question posts for each programming language posted in stack Overflow between the period of 2012 to 2016 which had a score of greater than 1. This was done to ensure that these posts have been in the Stack Overflow for a longer period of time and have been properly reviewed and edited as required by the standards of the forum. The score of more than one ensure that its a good quality post. This strategy resulted in a dataset of 200,000 question posts.

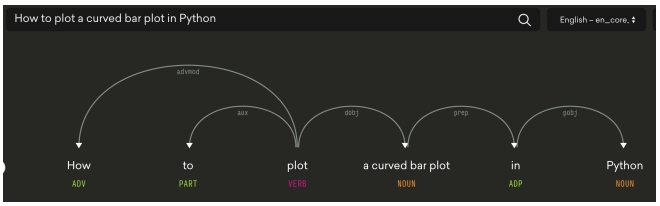


Fig. 3: An example showing how Spacy performs dependency parsing using displaCy dependency visualizer. We extracted all the nouns, proper nouns from the text after dependency parsing. This process removes noise and stores the most relevant information from each posts which could help machine learning models learn from the vocabulary and make better predictions.

The data in .xml was parsed using xmltodict and Beautiful Soup library in python to extract the code and text from the question posts separately. A Stack Overflow question post consists of a title, body and embedded code. The texts from the post was separated into three categorized namely-title alone, body alone and title and body alone. This was mainly done to check if title, body or the combined text could provide the highest accuracy in our classification task. The three categories of textual data was stored in separate SQL tables. The extracted code from the posts were not preprocessed and was stored in a SQL table.

The texts from the post (title and body texts) were preprocessed by removing all non English characters using Regex. Then we performed stemming and lemmatization using NLTK library in python. We created a custom stop word dictionary by combining the stop words from scikit learn[13], Spacy and NLTK [12].python libraries and these stop words were removed from both title and body text. The processed title and body text was stored into separate SQL tables. The combined title and body texts from posts were subjected to dependency parsing using the Spacy Library. After the dependency parsing we only extracted the named entities which are nothing but nouns or noun phrases.An example of how dependency parsing works is shown under figure (Fig3). This extracted vocabulary is really condensed and no longer subjected to any other processing and finally stored into another SQL table. Therefore at the end of this stage, we are left with a database with below 4 SQL tables.

- 200,000 Code snippets.
- 200,000 text snippets from SO post title
- 200,000 text snippets from SO post body
- 200,000 text snippets from SO post title + body

In order to answer our third research question about tools perform across various standards of code, We selected Python, C++ and Java as programming languages and its corresponding versions/standards. For python we extracted code for the tags: python-3.x, python-2.7, python-3.5, python-2.x, python-3.6, python-3.3 and python-2.6, for Java we selected java-8 and java-7 as the tags and finally for C++ we selected c++11, c++03, c++98 and c++14 as tags. We did not extract

the texts for these tags but only the code snippets. These code snippets were stored in a separate file folder.

IV. Methodology

In this section we talk in detail about the methodology we followed in answering our research questions. We used the tool from Algorithmia to predict the programming languages from the code snippets that we extracted from Stack Overflow. We selected 75 code snippets randomly from 5000 code snippets for each programming language. This created a subset of 1500 code snippets which could be used for prediction using Algorithmia. We used the urlib library in python to make the API calls to Algorithmia tool to generate predictions for all 1500 code snippets. The API call returns a JSON file with languages as key and corresponding probability for all 21 languages that it supported. We selected the language with the highest probability as the predicted language. We compared the predicted language with the ground truth to generate all our metrics.Our Metrics included accuracy, Precision, recall and F score. We also created a confusion matrix to further evaluate our results.

In order for us to answer our second research question, we divided our code snippets into three categories.

- Lines of Code $<$ 12 : Small code snippet
- Lines of Code $>$ 12 and $<$ 24 : Medium code snippet
- Lines of Code $>$ 24 : Large code snippet

We selected 25 Code snippets from small, medium and large category for each programming language and generated predictions. We then compared the lines of code with the accuracy of the tool. We also wanted to study if there is any correlation between the lines of code and prediction accuracy of the tool. We created two numpy arrays, one with the predictions (1 if correct, 0 if wrong) and other with the lines of code for all 1500 code snippets that we predicted using Algorithmia. We used the Pearson correlation from the Scipy stats library in python to compute the correlation between the two arrays. In Pearson correlation a positive score indicate a positive correlation between two entities you are comparing and a negative score indicates negative correlation. This is discussed in detail under results and discussion section of this paper.

The next phase of our research involved predicting languages from processed text using Machine learning. The texts from the posts were split using Tf-Idf vectorizer from the Scikit learn library. we set the min-df (minimum Document frequency) parameter to ten which means that we will only include words which are present in at least 10 documents among all available documents. This scheme helps us to remove the rare words from our dataset and this in turn reduces the noise. We left the max-df parameter to default since we have already removed the stop words in the data pre-processing step in section 3. We selected three machine learning classifiers for our prediction task - Random Forest Classifier, MultiNomial Naive Bayes and SVM with Rbf kernal. However, The SVM timed out due to computational cost and time and therefore not reported in this paper. We report precision, recall, Accuracy, F score and confusion matrix as our metrics for both the classifiers in section 5.

The machine learning models were hypertuned using GridSearchCV - A tool for parameter search in Scikit-learn. For a multinomial Naive Bayes classifier, it is important to tune a hyper parameter called alpha (Additive smoothing) parameter.Similarly, Random Forest Classifier which is an bagging classifier has a parameter called 'no of estimators' which are subtrees used to fit the model. These parameters were fixed after performing GridSearch on the Cross validation sets (10 fold cross validation). It may be noted that the dataset was split into train and test in the ratio 80:20, where train was used for training and cross validation and test set was used to report all final metrics in this paper.

The Machine learning method described above was repeated for title alone, body alone and title + body text categories of data and reported under results and discussion. We then compared the confusion matrix from best machine learning scheme from texts with that of the confusion matrix from predicting language from code snippets.

V. Results and Discussions

We begin by evaluating the results we obtained by running the prediction of programming languages using the tool from Algorithmia for 1500 snippets of code across 20 different programming languages.

RQ 1. *Can we predict the tags of programming languages using code embedded in the Stack Overflow post using existing tools?*

The tool gave an accuracy of 62.8% in predicting languages from code snippets with an F score of 0.64. The tool has a precision of 0.68 and a Recall of 0.63. The confusion matrix for this prediction task is shown in Fig5. We can infer from the confusion matrix that some languages such as Scala, Haskell, Ruby, Objective C and python performs much better as compared to languages languages such as HTML, CSS, JavaScript and SQL performs the worst. Scala and Haskell are not often confused with other languages due their unique syntax. It is bit surprising to us that Objective C is not confused with C language since they both have very similar syntax. It could be because objective C is much more similar to C++ than C language. This is evident from some Objective C code snippets which are confused in C++ in the Fig5.

HTML, CSS and JavaScript are used together in web development and moreover, CSS and JavaScript are often embedded in the HTML files which could explain why the tool performed very poorly in identifying these languages. We manually analyzed the Code snippets from HTML, CSS

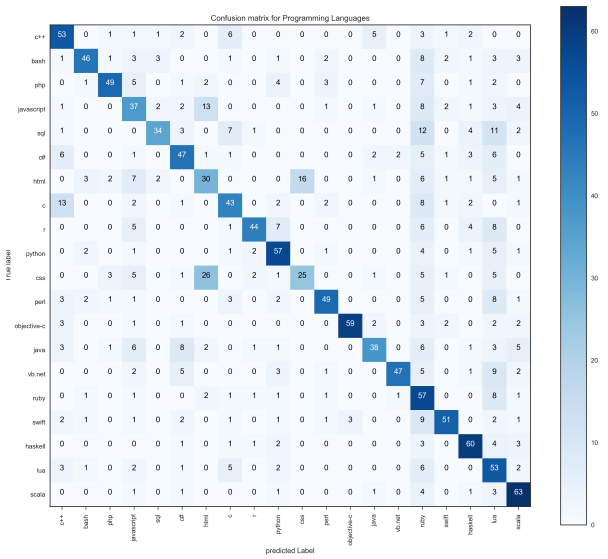


Fig. 4: The Confusion matrix for predicting programming. We used 75 code snippets for each programing language. A prominent diagonal indicates that the tool is the results of the test are above the chance accuracy which is 5%.

and JavaScript and found that in more than 60% of the cases, the tool generated wrong predictions because of them being used together in the same code snippet. However, in rest of the cases the tool simply got them wrong. We speculate that this could be due to the reason that training data might have had embedded CSS and JavaScript in them which makes the tools performance on these languages much lower as compared to other languages. The poor performance of SQL could be attributed to the fact that SQL is mostly embedded in the code snippets of other programming languages such as Python, C, C++, Java etc. From the confusion matrix, we could see that SQL is mostly confused with Scala, Ruby and R. It seems the tools confused languages with languages with the same mean lines of Code. The mean and median lines of code for each 20 programming language extracted from over 200,000 code snippets are shown in Fig6.

RQ 2. *How does the lines of code impact the prediction capability of tools which predict programming languages from code snippets?*

Before we answered the above question, we checked if there is any correlation between lines of code and the number of characters per code snippets across all programming language. The usual assumption is that complexity of the code mostly depends upon the lines of code. Another assumption is that there is always a strong correlation between lines of code and number of characters in code. We tested this assumption using Pearson correlation and found a very high positive correlation of +0.8886. We also performed correlation analysis between loc and no of characters in code snippets for all 20 languages separately and found that they all had strong positive correlation. Bash had the

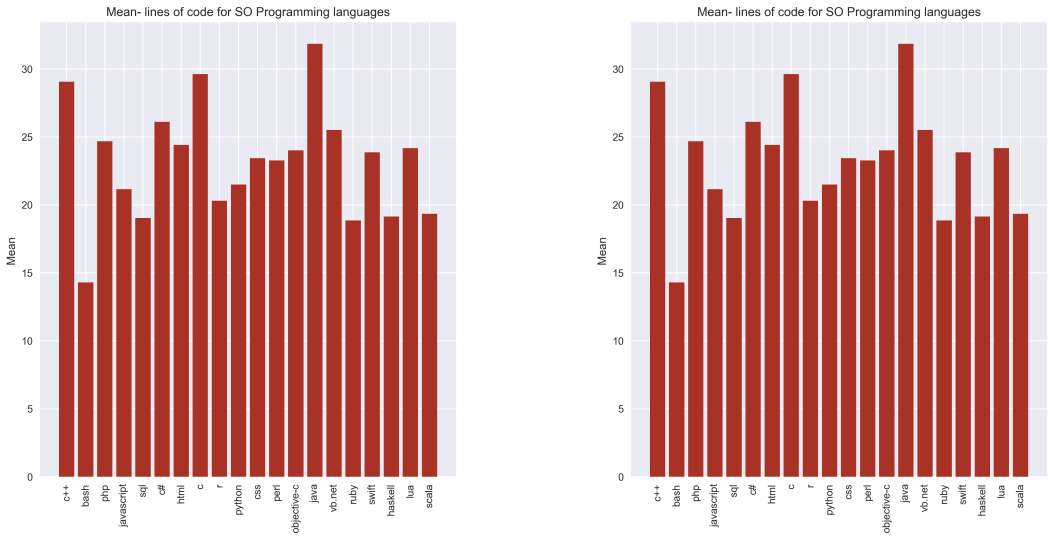
Language	Precision	Recall	F1-score
objective-c	0.95	0.79	0.86
haskell	0.72	0.8	0.76
vb.net	0.94	0.63	0.75
scala	0.68	0.84	0.75
php	0.84	0.65	0.74
swift	0.82	0.68	0.74
perl	0.8	0.65	0.72
python	0.67	0.76	0.71
bash	0.81	0.61	0.7
r	0.85	0.59	0.69
c++	0.6	0.71	0.65
c#	0.62	0.63	0.62
java	0.75	0.51	0.6
c	0.61	0.57	0.59
sql	0.81	0.45	0.58
lua	0.38	0.71	0.49
javascript	0.45	0.49	0.47
ruby	0.34	0.76	0.47
css	0.61	0.33	0.43
html	0.39	0.4	0.4
avg / total	0.68	0.63	0.64

Fig. 5: Precision, Recall and F score metrics for all languages predicted by Algorithmia.

lowest correlation but still significant positive correlation of +0.775 and Lua had the highest correlation of +0.987. Thus we conclude that using lines of code to test the relationship between complexity and prediction capability of the tool is appropriate and equivalent to using number of characters.

We made predictions using small, medium and large snippets of code separately for each of the 20 programming languages using Algorithmia. In each case we had 25 code snippets from every language. We found that the tool accuracy with small snippet of code was 40.6%, medium snippet 70.8% and large snippet 71.6%. We therefore conclude from this that tools does not perform well with very small snippets of code (Less than 12 lines of code). We could assume that the tool was never trained on small snippets of code and therefore it couldn't make predictions on similar size code snippets. We also performed another correlation analysis between the prediction of the tool (1 being correct language predicted and 0 as incorrect language predicted) and the corresponding lines of code for each of 20 programming languages. The result is summarized by the figure Fig9

As we can see from the Fig 7 that, there does exist a weak to medium positive correlation between the ability to predict the language and the lines of code of the code snippet given as input to the tool. This correlation is not the same for all



(a) Mean LOC for 20 languages

(b) Median LOC for 20 languages

Fig. 6: A bar plot showing mean and median number of lines of code for 20 programming languages. Around 200,000 Code snippets were extracted from Stack Overflow Question posts and used to perform this analysis. SO recommends users to include complete pieces of code to enable the question posts to receiver faster and better answers to the questions.

the programming languages. We found zero correlation for Perl and weak negative correlation for HTML, CSS and R. This means that as the line of lines of code increases, the capability of the tool to predict these languages drops. For R, it might b due to its complex syntax and structure. It could be worth investing more time in studying the specific functions or syntax which are used in these languages that might be potentially confusing the classifier/tool.

RQ 3. *How does the tools that predict programming languages from code snippets perform across various standards of the same language?*

We extracted code snippets from Stack Overflow posts with tags on versions/standards of Java, Python and C++ as described under methodology. We extracted the prediction using Algorithmia and found that it achieved an accuracy of 88% for all versions of C++, for Java 60% and finally for python it was 80.6%. The scores seems to be similar to the one which we obtained for Research Question 1 for these languages. However, we could conclude that the tools does not in fact generalize in predicting across various versions or standards of the language.

RQ 4. *Can we predict the tags of programming languages by analyzing the texts describing a stack Overflow question?*

We divided this task into three phases: Predicting the languages with just the title of the post, body of the post and with the named entities extracted after performing dependency parsing of both title and text combined. The results of all our machine learning models for all the three phases are

Classifier	Data	Accuracy
MultiNomial Naïve bayes	Title only	66.50%
Random Forest	Title only	61.57%
MultiNomial Naïve bayes	Body only	62.90%
Random Forest	Body only	61.53%
MultiNomial Naïve bayes	Title + Body	68.10%
Random Forest	Title + Body	69.35%

Fig. 7: Table showing the results of Machine learning on the text. We can see from the table that Combined title and body results in a better classifier than the classifiers trained on title or body alone.

summarized by the Fig7. The best performing classifier was a Random Forest Classifier with 100 number of forests as estimators. The model was trained on 150,000 text snippets from SO posts (Title and body combined). The process of extracting only Named entities using dependency parsing seems to have reduced the noise resulting in higher accuracy. It is worth noticing that the title of the post has higher information than the body of the post (Based on classifier results).

We also plotted a confusion matrix for the NLP task above. The accuracy from using NLP is higher 69.35% as compared to 62.8% accuracy using Algorithmia. The precision, recall and F Score are 0.682, 0.67 and 0.668 respectively. These metrics are much better as compared to the ones from Algorithmia. The confusion matrix is shown under Fig8. This confusion matrix shown an entirely different story as compared to that of one with code snippets. SQL seems

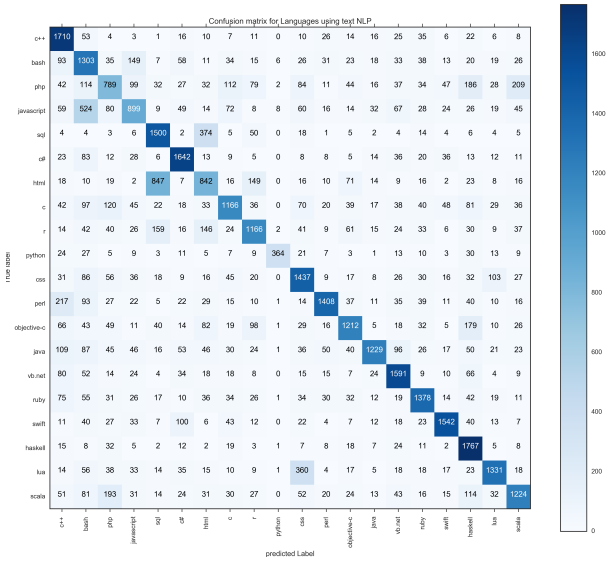


Fig. 8: Confusion matrix for language prediction from text using random forest classifier.

to be most confused with HTML, Bash is confused with JavaScript and the worst performing language is python. C++, Haskell, C# and vb.net seems to be top performers. The confusion between HTML, CSS and JavaScript is moderate as compared to that of Algorithmia. The main conclusion we would like to make is that some languages are predicted with better accuracy with text and some with source code. Therefore training a meta classifier which uses both code and text as input would give us better prediction capability. Unfortunately, such method is beyond the scope of this paper.

VI. Future Work

We have identified lack of tools in predicting languages from source code. Most of the existing tools focus on file extension rather than the code itself in predicting the language. We evaluated the tool Agorithmia and found it insufficient in predicting the languages. The main reason we believe for the poor performance of the tool was the lack of a standard dataset for training. We need a dataset which takes into account various complexities of the languages such as line of code, characters, with and without library imports, different function calls, various standards and versions. We believe that Stack Overflow would be the perfect source for such a dataset. As we discussed in Introduction section of this paper, the Stack Overflow is one of the biggest source of source code. We are confident that a dataset build from Stack Overflow would be most appropriate for training tools.

We propose extracting 10000 code snippets across 20 programming languages from Stack Overflow. This dataset should also include various standards of languages described in the paper and accommodate various size of code snippets (Small, Medium and large). We would remove embedded SQL from all programming language snippets and for SQL, we wound include both embedded and free SQL. For HTML, CSS and JavaScript, we would ensure that they are kept separate from one another. This is because a probabilistic based tool could be able to identify all the three languages in a single embedded post if trained with noise free data.

There have been tremendous progress made in the field of deep learning especially time series or sequence based models such as RNN (Recurrent Neural Networks) and LSTM (Long short term memory) Networks. These networks could take a sequence of text basically character by character and could predict the category of the sequence. So here we could give the source code one character at a time as input to these RNN and LSTM models and we would be able to predict the target programming language. One small draw back of this methodology is that its computationally very expensive and might need large GPU cluster to train and test.

Natural Language processing and machine learning techniques performed much better in predicting languages as compared to tools which predict from source code. Therefore, we need to ensure that additional research should be carried out to improve the results of this task. Stack Overflow texts are pretty unique in the sense they capture both the tone, sentiments and vocabulary of the developer community. There might be a variation between this vocabulary for each programming language. Therefore, it is important that we capture, understand and separate the vocabulary for each programming language. This could be achieved through techniques such as LDA which would predict the probability that a word belong to a particular language or word2vec which gives the vector distance of a language from other languages (based on its vocabulary). This kind of analysis not only would be useful in identifying languages, but also would enable us to find languages which are close to one another. We propose the creation of a dictionary of named entities unique to each of the 20 programming languages. We believe such a dictionary would help the researchers in creating better machine learning and NLP models to predict languages from their documentation or descriptions such as one in Stack Overflow.

This paper only explored three machine learning classifiers - Random Forest Classifier, SVM and MultiNomial Naive Bayes. There are classifiers such as AdaBoost, XGBoost, LDA classifier which might help to improve the results. We also recommend training a CNN network (Convolutional Neural Network) which are used along with Word2Vec for various NLP related tasks. CNN have shown considerable improvement in document classification tasks as compared to most simple classifiers.

VII. Threats to Validity

Internal validity: We have only studied the effect of parameters such as lines of code, programming language versions on the capability of the tools. There might be other factors which might affect the complexity of source code and in turn the accuracy of the tools. The use of dependency parsing could result in loss of important vocabulary which might affects the results of our experiments with texts. However,

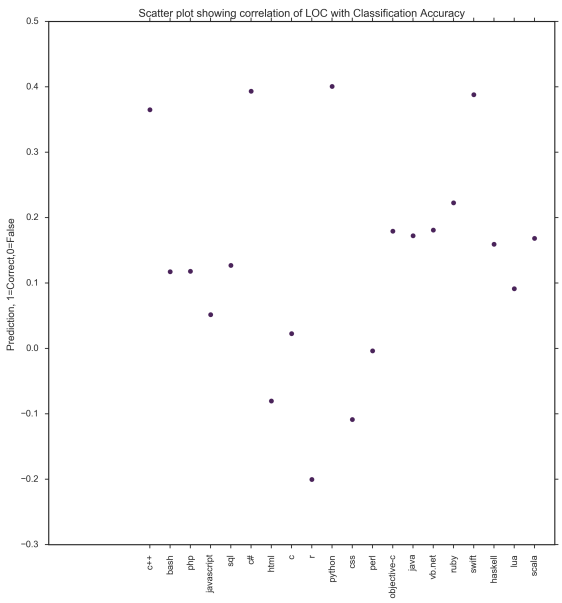


Fig. 9: Scatter plot showing correlation between language prediction capability and the lines of code for 20 programming languages.

we have manually analyzed the vocabulary before and after the dependency parsing and other text processing steps to ensure that information related to the languages are not lost.

External validity: We have only used Stack Overflow as the source of data for our analysis. We have not explored other source such as GitHub repositories or other sources of code. Therefore, we cannot be absolutely confident that our results would be the same across all the sources of code snippets and texts/documents on programming languages. Our research also only focused on one tool for answering our research questions. This is mainly due to the lack of availability of open source tools for predicting languages. Therefore, there could be other tools which might have better capability in predicting from code snippets. We have tried our best to include almost all the most commonly used programming languages for our study. However, there are still more popular and powerful languages such as Cobol, Go, pascal etc which were not considered for this study.

VIII. CONCLUSIONS

In this paper we discussed in brief about the importance of predicting languages from source code and textual data. We argued that the existing tools for predicting the programming language from source code are not adequate enough considering the challenges and complexity of today's programming languages. We evaluated a tool called ProgrammingLanguageIdentification by Algorithmia market place which is a paid API service and found that it achieved only an accuracy of 62.8% on code snippets from Stack Overflow posts. This is in direct contrast with the reported accuracy of the tool which is 99.4%. The found that some languages such as Objective C, Haskell etc. performed better as compared to languages such as HTML, CSS and JavaScript which are often used together. We created and analyzed the confusion matrix from our prediction results to identify languages which are either similar or used together. For example, in Stack Overflow SQL is often used as embedded code in other languages such as C, Java, C++ and python.

Similarly we studied the impact of lines of code on the prediction accuracy of the tool and found that there is a positive correlation between them. However, there was an exception in case of R, HTML and CSS where the prediction capability of the tool decreased with increase in lines of code. We argued that accuracy of the tool that we obtained using our custom dataset from Stack Overflow posts is much lower than the claimed accuracy of the tool due to the fact that the tool was trained and tested on GitHub repositories which are generally source code files with large number of lines of code. This makes the tool not efficient in predicting small code snippets, code snippets with embedded SQL or HTML files with CSS and JavaScript code embedded in them. The tool also failed to classify code snippets across various standards/versions of the same programming language such as c++11, c++03, c++98 and c++14 for C++. We speculated that this could be due to the fact that the tool was only trained on one standard/version of the language. The tool seemed to generalize better for various C++ standards but performs poorly across standards of Java and Python.

We also used the texts from the posts in Stack Overflow posts to predict programming languages and found a better accuracy of 69.4% as compared to 62.8% accuracy from predicting using code snippets. We also compared the confusion matrix from the prediction using code snippets with that of one from text using Machine Learning and found that they are entirely different. This means that results from text and code snippets could be combined to create better prediction capabilities (Meta Classifier).

We strongly believe that more research needs to be done in creating tools for predicting programming languages from code snippets. We find that the existing tools are not sufficient enough to accommodate all the complexities of today's languages. We strongly recommend for the creation of a standard dataset (with text and code snippets) which could be used to benchmark and evaluate all the the existing and future tools for predicting programming languages

APPENDIX

This section is meant only for the purpose of evaluation for the MSR course. The authors would like to indicate all the work done in this project was shared equally which includes the writing of this report. The work related to research question one and three was conducted by Kamal. The work related to research question 2,4 which includes Machine learning, Natural Language processing and Correlations were done by Dhanush Dharmaretnam. We request you to get in touch with us for any clarifications.

ACKNOWLEDGMENT

The authors would like to thank Daniel German for his guidance and advices during this work. Kamel Alrashedy acknowledges the financial support of the Saudi Ministry of Education through a graduate scholarship. The authors would also like to thank the Compute Canada and WestGrid Consortium for the computation resources provided for various data processing and machine learning tasks.

REFERENCES

[1] A. K. Saha, R. K. Saha, and K. A. Schneider, A discriminative model approach for suggesting tags automatically for stack overflow questions, in Proceedings of the International Workshop on Mining Software Repositories. IEEE Press, 2013, pp. 7376.

[2] C. Stanley and M. D. Byrne. Predicting Tags for StackOverflow Posts. In Proceedings of ICCM 2013 (12th International Conference on Cognitive Modeling), 2013.

[3] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns, What makes a good code example? A study of programming Q&A in StackOverflow, in Proc. ICSM, pages 2535, 2012.

[4] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, Two Studies of Opportunistic Programming: Interleaving Web Foraging, Learning, and Writing Code, in Proceedings of CHI 2009, New York, NY, USA, 2009, pp. 15891598.

[5] C. Treude, O. Barzilay, and M.-A. Storey, How Do Programmers Ask and Answer Questions on the Web? in Proceedings of ICSE 2011, New York, NY, USA, 2011, pp. 804807.

[6] C. McMillan, M. Grechanik, D. Poshyvanyk, Q. Xie, and C. Fu, Portfolio: A Search Engine for Finding Functions and Their Usages, in Proceedings of ICSE 2011, 2011, pp. 10431045.

[7] M. Revelle, B. Dit, and D. Poshyvanyk, Using Data Fusion and Web Mining to Support Feature Location in Software, in Proceedings of ICPC 2010, 2010, pp. 1423.

[8] R. Holmes, R. J. Walker, and G. C. Murphy, Strathcona Example Recommendation Tool, in ACM SIGSOFT Software Engineering Notes, New York, NY, USA, 2005, pp. 237240.

[9] B. Seaman, The Information Gathering Strategies of Software Maintainers, in Proceedings of ICSM 2002, 2002, pp. 141 149.

[10] V. S. Rekha, N. Divya, and P. S. Bagavathi. A hybrid auto-tagging system for stackoverflow forum questions. In Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing, ICONIAAC 14, pages 56:156:5, New York, NY, USA, 2014. ACM.

[11] Algorithmia, ProgrammingLanguageIdentification tool", 2017. [Online]. Available: https://www.algorithmia.com.

[12] E.Loper, S. Bird, NLTK: The natural language toolkit, in Proc. Interact. Present. Sessions Association for Computational Linguistics, 2006, pp. 6972.

[13] F.Pedregosa,G.Varoquaux,A.Gramfort,V.Michel,B.Thirion,O.Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research, 2011.

[14] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, Design Lessons from the Fastest Q&A Site in the West, in Proceedings of the 2011 annual conference on Human factors in computing systems, New York, NY, USA, 2011, pp. 28572866.

[15] M. Asaduzzaman, A. S. Mashiyat, C. K. Roy, and K. A. Schneider. Answering questions about unanswered questions of stack overflow. In MSR, 2013.

[16] SoStats, 2017. [Online]. Available: https://sostats.github.io/last30days/

[17] Quantcast, 2017. [Online]. Available: https://www.quantcast.com