Predicting the Programming Language of Questions and Snippets of Stack Overflow Using Natural Language Processing

by

Kamel Alrashedy
B.Ed., University of Hail, 2013

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Predicting the Programming Language of Questions and Snippets of Stack Overflow Using Natural Language Processing

by

Kamel Alrashedy
B.Ed., University of Hail, 2013

Supervisory Committee

Dr. Venkatesh Srinivasan, Co-Supervisor
(Department of Computer Science)

Dr. T. Aaron Gulliver, Co-Supervisor
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Venkatesh Srinivasan, Co-Supervisor

(Department of Computer Science)

Dr. T. Aaron Gulliver, Co-Supervisor

(Department of Electrical and Computer Engineering)

ABSTRACT

Stack Overflow is the most popular Q&A website among software developers. As a platform for knowledge sharing and acquisition, the questions posted in Stack Overflow usually contain a code snippet. Stack Overflow relies on users to properly tag the programming language of a question and assumes that the programming language of the snippets inside a question is the same as the tag of the question itself. In this thesis, a classifier is proposed to predict the programming language of questions posted in Stack Overflow using Natural Language Processing (NLP) and Machine Learning (ML). The classifier achieves an accuracy of 91.1% in predicting the 24 most popular programming languages by combining features from the title, body and 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 81.1%. Finally, we propose a classifier of code snippets only that achieves an accuracy of 77.7%.Thus, deploying ML techniques on the combination of text and code snippets of a question provides the best performance. These results demonstrate that it is possible to identify the programming language of a snippet of only a few lines of source code. We visualize the feature space of two programming languages Java and SQL in order to identify some properties of the

information inside the questions corresponding to these languages.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

First of all, I would like to thank my co-supervisor, Venkatesh Srinivasan, for his guidance and advice throughout my research. The opportunities he gave me to work on many projects expanded my knowledge and led me to interest in Applied Machine Learning area. Frequent discussions with Venkatesh about my progress had a positive impact and motivated me to learn more. Beside research experience, I have learnt many things from him such as communication skills, time management, facing challenges, planning for the future, work-life balance, learning from failure and managing failure. He was an extremely helpful supervisor.

I would like to thank my co-supervisor, Aaron Gulliver, for guidance during my masters degree. His motivation and enthusiasm led me to work on the research problem in this thesis. His expectations from my work had a positive impact and made me work hard. Also, working on another project with him ended up as my first published paper. I learnt the process of doing research and publishing a scientific paper from him. I respect that he motivated and helped me to start my degree at this university.

In addition, I would like to thank, Daniel German, for my favorite course Mining Software Repository that highly influenced my current and future research work. Collaborating with him was an extremely great opportunity. I have received many comments and suggestions from him on how to improve this work. Dhanush Dharmaretnam, my classmate, officemate and friend, helped and supported me in working in the field of Nature Language Processing to solve the main problem considered in this thesis.

I would like thank many great friends and colleagues who encouraged and motivated me during this time. Special thanks to Rehan Sayeed for his encouragement and enthusiasm that motivated me a lot during my masters degree. I acknowledges the financial support of the Saudi Ministry of Education through a graduate scholarship.

DEDICATION

First and foremost, I would like to thank almighty God without whose mercy this journey was near impossible to voyage.

I would like to dedicate this thesis to my wonderful parents, Shima and Aali, whose constant encouragement throughout my journey has been really awe aspiring. This journey would remain incomplete without my aunt Shima whose frequent phone calls and emails were the source of mental strength needed to complete my masters.

I would also like to dedicate this work to my aunt, Maha, who is the most influential person throughout my education career. It was her that initially persuaded me to pursue a graduate degree.

I cannot forget to thank my loving brothers Wafi and Ahmad for encouraging me to study Computer Science. Many thanks to my brother Ayyad, my first programming teacher, who taught me HTML and Microsoft FrontPage when I was in grade nine.

Finally, I would also like to thank my trainer, Yaser, who taught me programming languages and basics of software engineering. Many thanks to Yaser for teaching, guidance, encouragement and motivation during my undergraduate years.

"I have not failed. I've just found 10,000 ways that won't work."

"Genius is one percent inspiration and ninety-nine percent perspiration."

Thomas Edison

Chapter 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. Along with the growth of Stack Overflow, the number of programming languages in use has increased. In its 2018 Developer's Survey, Stack Overflow lists 38 different programming languages in its list of most loved, dreaded and wanted languages. The TIOBE Programming Language index tracks more than 100 languages [2].

Forums like Stack Overflow 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.

Another related problem is the identification of the programming language of a snippet of code. Given a few lines of code, it is often necessary to identify the language in which they are written. Stack Overflow relies on the tag of a question to determine how to typeset any snippet in it. If a question is not tagged with any programming language, then the code is not typeset; however, the moment a tag is added, the code is rendered with different colours for the different syntactic constructs of the language. If a question has snippets in two or more languages, Stack Overflow will only use the first programming language tag to typeset all the snippets of the question.

The problem of identifying the language of snippets spans beyond Stack Overflow. Snippets are widely included in blog posts and stored in cut-and-paste tools (such as Github Gists[1] and Pastebin[2]). They might also be stored locally by the user in tools such as QSnippets or Dash. Gists require the user to give each snippet a filename, which it uses to classify the programming language. Pastebin, like most snippet management tools, requires the user to select the language of the snippet manually. In both cases, the onus is on the developer to specify the language. Most tools that use the source code in documentation (such as recommenders) typically require that the document is already tagged with a programming language to be processed (and assumes all the snippets in it are written in that language).

There is a substantial amount of textual information available about programming languages that is written in natural languages to describe or discuss either code snippets or the programming language. Textual information can be in the form of a blog, code documentation, bug report, code comment, or a summary of a code snippet. Websites such as Github[3], Jira[4], bugZilla[5], Stack Overflow[6] and Quora[7] contain textual information that explain or discuss code snippets or programming languages. Stack Overflow posts are an example of textual information about programming lan-

[1]https://gist.github.com/
[2]https://pastebin.com/
[3]https://github.com/
[4]www.atlassian.com/Jira
[5]https://www.bugzilla.org/
[6]https://stackoverflow.com/
[7]https://www.quora.com/

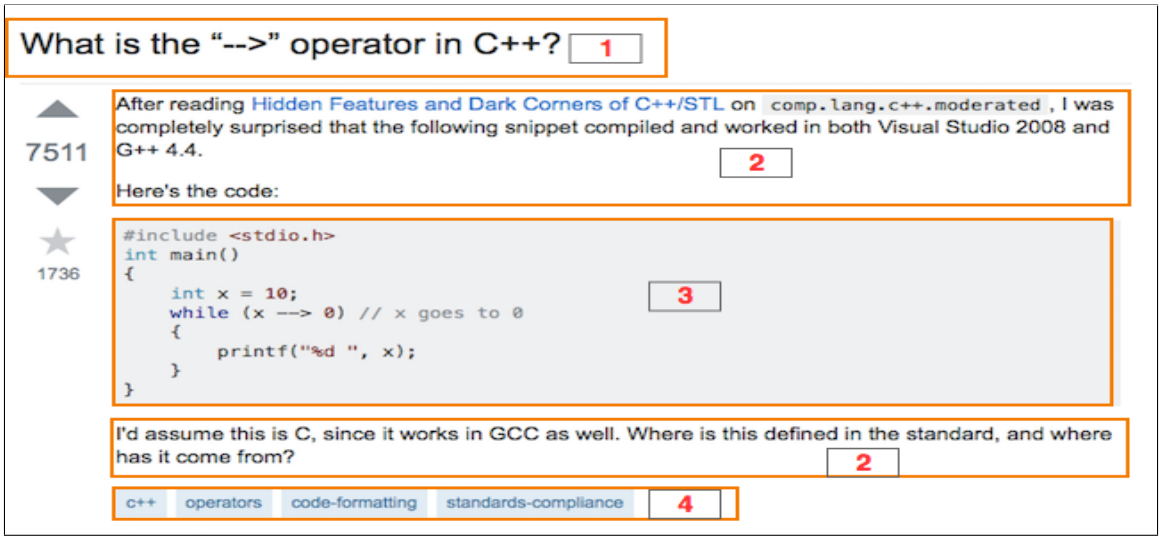


Figure 1.1: An example of a stack overflow post.

guages. Most Stack Overflow posts contain a code snippet and textual information or only textual information. This information could be mined to improve programming language prediction. In fact, it could be used to predict the language without using the code snippet.

In Stack Overflow, both the title and text body are examples of textual information. Users post a question to find a solution to their issues or seek knowledge related to a particular programming language. The post might only contain textual information without a code snippet. Analyzing the textual information to predict the programming language is proposed in this research.

Fig. 1.1 shows an example of a Stack Overflow post[8], (1) is the title (2) is the text body where users write, (3) is the code snippet, and (4) is post tags. Users are allowed up to 5 tags.

In this thesis, Machine Learning (ML) and Natural Languages Processing (NLP) models are used to predict the programming languages in Stack Overflow questions from textual information, code snippets and the combination of textual information and code snippets.

[8]https://stackoverflow.com/questions/1642028/

1.1 Research Questions

In the previous section, the importance and motivation of predicting the programming languages was discussed. The investigation of this problem leads to three main research questions.

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 (textual information) and code snippets in it. The purpose of the second research question is to determine if the inclusion of code snippets is an essential factor to determine the programming language that a question refers to. And 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.

1.2 Thesis Contributions

The contributions of the thesis 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 91.1%, precision is 0.91 and recall 0.91 in predicting the tag of a programming language.

2. A classifier that uses only textual information in Stack Overflow questions to predict their programming language. This classifier achieves an accuracy of 81.1%, a precision of 0.83 and a recall of 0.81 which is much higher than the previous best model (Baquero *et al.* [4]).

3. A prediction model based on Random Forest [8] and XGBoost [6] 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 77.7%, a precision of 0.79 and a recall of 0.77.

4. Use of Word2Vec [18] to study the features of two programming languages, Java and SQL. These features are projected into a 300 dimensional vector space using Word2Vec and visualized using T-SNE.

1.3 Thesis Organization

This section presents the organization of this thesis. It contains seven chapters which are briefly described below.

Chapter 1 contains the introduction, a statement of the claims which will be prove in the thesis, and an overview of the structure of the thesis.

Chapter 2 discusses the related work on predicting the programming languages from source code files and predicting the question tag of Stack Overflow. It also gives the main motivation behind this thesis.

Chapter 3 discusses the reasons behind selecting the Stack Overflow dataset for this study. It also presents how the dataset was extracted from Stack Overflow, and the cleaning and processing of the textual information (body and title) using natural languages processing techniques.

Chapter 4 gives detailed information regarding the experiments used to mine the question post from Stack Overflow. It provides details about the machine learning algorithms that used in this study.

Chapter 5 presents the results of the proposed models for each research question. Detailed explanations are given of the most important observations for the various programming languages.

Chapter 6 presents a discussion of the results followed by a comparison of using textual information and code snippets datasets. It also discusses the visualization of features extracted from Stack Overflow datasets followed by threats to the validity of this work.

Chapter 7 gives the conclusions of this thesis and future research directions in predicting the programming languages.

Chapter 2

Related Work

2.1 Predicting Programming Languages

Baquero *et al.* [4] proposed a classifier to predict the programming language of a Stack Overflow question. They extracted a set of 18000 questions from Stack Overflow that contained text and code snippets, 1000 questions for each of 18 programming languages. They trained two classifiers using a Support Vector Machine model on two different datasets which are text body and code snippet features. The evaluation achieved an accuracy of 60% for text body features and 44% for code snippet features which are much lower than the results obtained in this thesis.

Kennedy *et al.* [12] studied the problem of using natural language identification to identify the programming language of entire source code files from GitHub (rather than questions from Stack Overflow). Their classifier is based on five statistical language models from NLP and identifies 19 programming languages and can achieve a high accuracy of 97.5%. In our work, 24 programming languages are predicted using code snippets rather than source code file. Similarly, Khasnabish *et al.* [13] proposed a model to detect 10 programming languages using source code files. Four algorithms were used to train and test the model using Bayesian learning techniques, i.e. NB, Bayesian Network (BN) and Multinomial Naive Bayes (MNB). It was shown that MNB provides the highest accuracy of 93.48%.

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 [16] 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.* [11] developed a tool called Strathcona that can find similar snippets of code.

Rekha *et al.* [23] 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.* [25] converted Stack Overflow questions into vectors, and then trained a Support Vector Machine using these vectors and suggested tags using the model obtained. The tag prediction accuracy with 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 [26] 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, the accuracy is only 65%.

2.2 Mining Stack Overflow

The Stack Overflow dataset is the most popular discussion forum among researchers to study and understand the discussion among developers. Barua *et al.* [5] analyzed the textual information in stack overflow posts to discover the main topics in the developer community by using Latent Dirichlet Allocation (LDA) and a statistical topic modeling technique. Similarly, Rosen *et al.* [24] analyzed questions that were

asked by the mobile developers. Treude *et al.* [27] studied how programmers ask and answer questions in social media to explore which questions received a good answer and why other questions are not answered, choosing Stack Overflow as a dataset to analyzing question and answer posts.

Morrison *et al.* [13] studied how the programming knowledge related to age and used the data from Stack Overflow to answer their research questions. To understand the factors that differentiate a good post from a bad post, Nachi *et al.* [19] studied what makes a post good or bad based on the number of lines of code, quality of code and texts in a Stack Overflow post. Similarly, Treude *et al.* [27] analyzed Stack Overflow questions to explore the type of questions that receive a good answer. They found that posts which contained snippets of code received good answers.

Another approach that studied answered and unanswered questions in Stack Overflow was by Denzil *et al.* [10]. In the last few years, there are many unanswered questions totaling 7.5% of Stack Overflow posts. They proposed a classifier to predict how long it takes for a question post to receive an answer. As Stack Overflow contains a large volume of questions, some users ask a question before they search in Stack Overflow. If their question already exists in Stack Overflow, it is a duplicated question. M. Ahasanuzzaman *et al.*[3] proposed a classification technique to mine duplicated questions. In [21] Gustavo *et al.* presented an empirical study to understand how software developers discuss energy-related questions in software community. The Stack Overflow dataset was used in this study and they analyzed more than 300 questions and 550 answer posts.

Chapter 3

Dataset Extraction and Processing

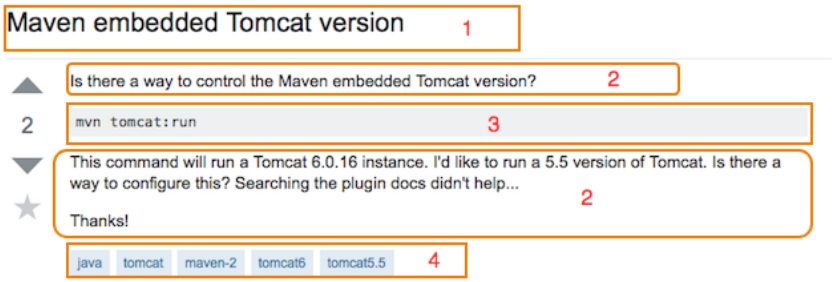
In this chapter, the details of the Stack Overflow dataset are discussed. Then, the preprocessing steps used for data extraction and processing are explained.

3.1 Stack Overflow Selection

Stack Overflow is the most popular web community for software developers to ask and answer questions. As of July 2017, Stack Overflow had 37.21 million posts, of which 14.45 million are questions with 50.9 thousand different tags. In this thesis, the programming language tags in Stack Overflow are of interest. The most popular 24 programming languages as per the 2017 Stack Overflow developer survey were selected for analysis [1]. They constitute about 93% of the questions in Stack Overflow. The languages selected for our study are: Assembly, C, C#, C++, CoffeeScript, Go, Groovy, Haskell, Java, JavaScript, Lua, MATLAB, Objective-C, Perl, PHP, Python, R, Ruby, Scala, SQL, Swift, TypeScript, Vb.Net and VBA.

3.2 Extraction and Processing of Stack Overflow Questions

The Stack Overflow July 2017 data dump was used for analysis. In this 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 10 characters. For each programming language 10,000 random questions were extracted. However, two programming languages had less than 10,000 questions: CoffeeScript (4,267) and Lua (8,460). The total number of questions selected was 232,727. Fig. 3.1(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 the text features during the training process to eliminate any inherent bias.



(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.

Figure 3.1: An example of a stack overflow question.

The .xml data was parsed using xmltodict and the Python Beautiful Soup library to extract the code snippets and text from each question separately as shown in Fig 3.2. A Stack Overflow question consists of a title, body and code snippet. In some

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

cases, a question contained multiple code snippets and these were combined into one. The questions were divided into three datasets: their titles, their bodies and their snippets. The title and body (referred to as textual information) and code snippet 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.

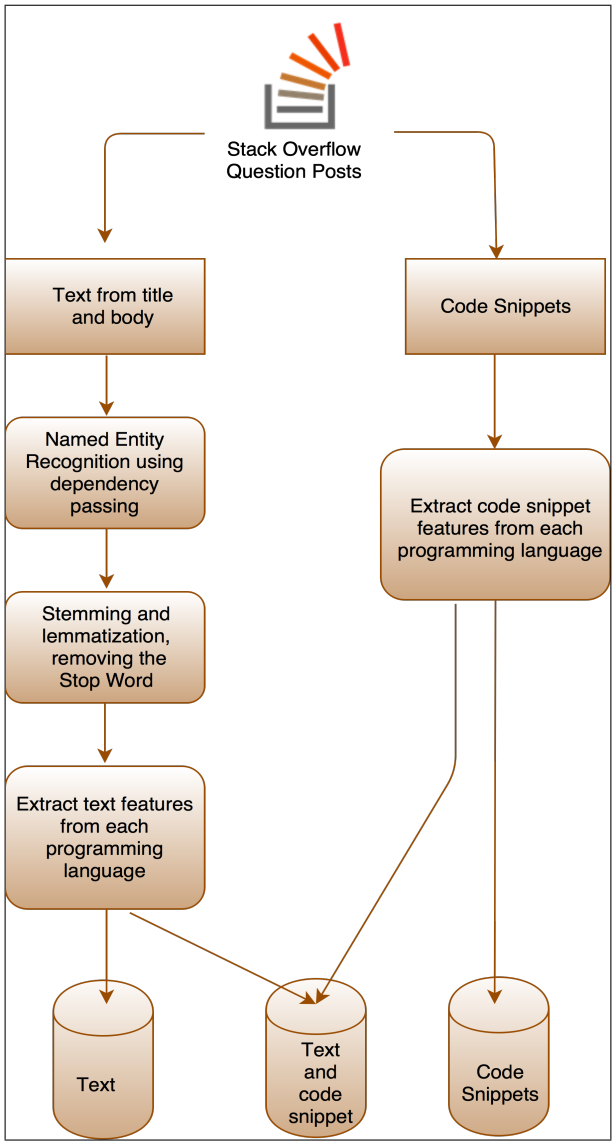


Figure 3.2: The dataset extraction process.

Machine learning models cannot be trained on raw text because their performance

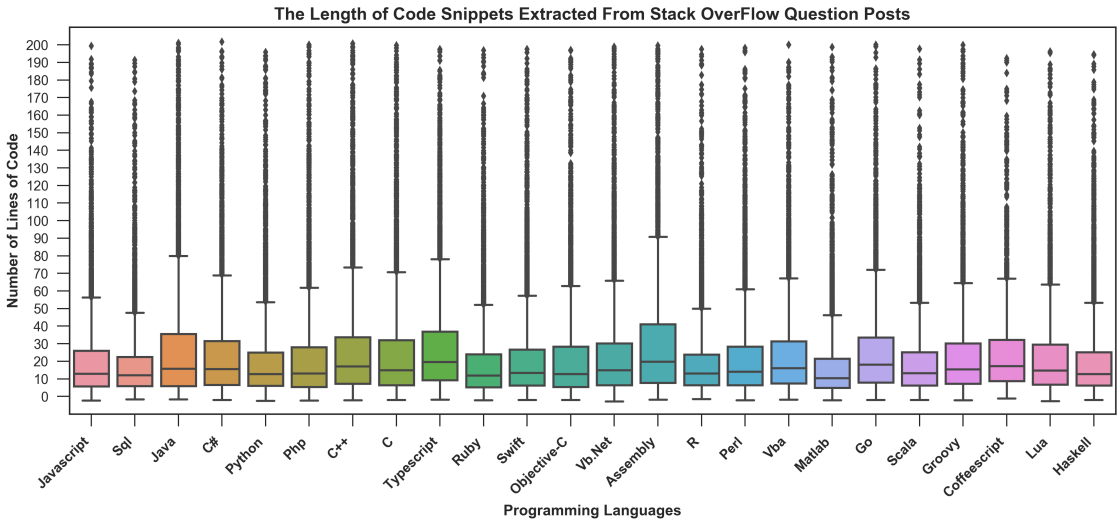


Figure 3.3: Box plots showing the number of lines of code in the extracted code snippets for all the languages. Note that there were at least 400 posts which had more than 200 lines of code and these were not included in this plot.

is affected by noise present in the data. The textual information (title and body) need to be cleaned and prepared before the machine learning model can be trained to provide a better prediction. Several preprocessing steps were required to clean the text. First, the non-alphanumeric characters such as punctuation, numbers and symbols were removed. Second, the entity names were identified using the dependency parsing of the Spacy Library [14]. An entity name is proper noun (for example, the name of an organization, company, library, function). Third, the stop words such as *after, about, all, and from* 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 [7]. At the end of all the preprocessing steps, the remaining words were used as features to help train machine learning model. Fig. 3.1(a) is the original Stack Overflow post and Fig. 3.1(b) is Stack Overflow post after application of NLP techniques.

The extracted set of questions provide a good coverage of different versions of a

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 the number of in lines of code as shown in Fig. 3.3. The number of lines of code in the snippets varied from 6 to 800.

Chapter 4

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 [20]. 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 Chapter 3.

4.1 Classifiers

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

Random Forest Classifier (RFC)

RFC [8] is an ensemble algorithm which combines more than one classifer. This classifier generates a number of decision trees from randomly selected subsets of 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 majority of trees. One advantage of this classifier is that if one or few of trees made noise or a wrong decision, it will not affect the accuracy of the result. Also, it avoids the overfitting problem seen in the Decision Tree model. The total number of trees in the forest is extremely important parameter because a large number of trees in the forest give high accuracy.

XGBoost

Extreme Gradient Boosting [9] (XGBoost) is a tree based model similar to Decision Tree and RFC. The idea behind boosting is to modify a 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 more intelligent 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 XGXoost uses a regularized model to help control overfitting and give a better performance.

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 is 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 datasets were split into training and test data using the ratio 80:20.

Another important contribution of this thesis is the study of the vocabulary and feature space for two programming languages (Java and SQL). A Word2Vec model [18] was used to visualize the features from code snippets and textual information (title and body) datasets using Gensim, which is a Python framework for vector space modelling [22]. The resulting model represented each word in the vocabulary in a 300 dimensional vector space. The selection of 300 as the number of dimensions for the trained word-vector is as per the recommendation of the original paper by T. Mikolov [17]. It is impossible to visualize concepts in such a large space, so T-SNE [15] was used to reduce the number of dimensions to 2. The most frequent 3% of words were selected from the vectors for 'Java and SQL' and analyzed using word similarity and cosine distance. The code snippets and textual information features which are close to each other in the vector space were selected for Java and SQL using the cosine distance. Then, these features were visualized to understand the similarities and differences between Java and SQL.

4.2 The Performance Metrics

In this section, the performance metrics of machine learning are presented. After the classifier learns from the extracted features, the classifier is tested on an unknown dataset to evaluate its performance. Each post in the testing dataset is included in the confusion matrix to describe how the posts are predicted. Precision measures how many predicted posts are relevant to a particular programming language. Recall measures how many relevant posts for a particular programming language are predicted. Java is taken as an example to illustrates the performances metrics.

True Positive: posts related to Java and predicted to be Java.

False Positive: posts not related to Java and predicted to be Java.

False Negative: posts related to Java but not predicted to be Java.

True Negative: posts not related to Java and not predicted to be Java.

$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$F1Score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Chapter 5

Results

In this chapter, the results obtained for the three research questions are described in detail.

5.1 XGBoost Classifier

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

To answer this question, the XGBoost and RFC classifiers were trained on the combination of textual information and code snippet datasets. The XGBoost classifier achieved an accuracy of 91.1%, and the average score for precision, recall and F1-score were 0.91, 0.91 and 0.91, respectively. In Table 5.1, the performance metrics for each programming language with respect to precision, recall and F1-score are given for XGBoost. Most programming languages have a high F1-score: Swift (0.97), GO (0.97), Groovy (0.97) and CoffeeScript (0.97) had the highest, while Java (0.75), SQL (0.78), C# (0.80) and Scala (0.88) had the lowest. Figure 5.4 shows the performance of the XGBoost classifier as a confusion matrix.

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

Programming	**Precision**	**Recall**	**F1-score**
Swift	0.98	0.96	0.97
Go	0.98	0.96	0.97
Groovy	0.99	0.95	0.97
CoffeeScript	0.98	0.96	0.97
JavaScript	0.97	0.95	0.96
C	0.98	0.95	0.96
C++	0.97	0.93	0.95
Objective-c	0.97	0.94	0.95
Assembly	0.96	0.95	0.95
Haskell	0.95	0.95	0.95
Python	0.97	0.91	0.94
Vb.Net	0.95	0.93	0.94
PHP	0.94	0.91	0.93
Ruby	0.89	0.93	0.91
Perl	0.91	0.91	0.91
MATLAB	0.92	0.90	0.91
R	0.91	0.89	0.90
Lua	0.94	0.86	0.90
TypeScript	0.90	0.88	0.89
VBA	0.85	0.91	0.88
Scala	0.85	0.92	0.88
C#	0.81	0.79	0.80
SQL	0.73	0.85	0.78
Java	0.70	0.82	0.75

Table 5.1: Performance of XGBoost trained on textual information and code snippet features.

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 81.1%, and the average score for precision, recall and F1-score were 0.83, 0.81 and 0.81 respectively. In Table 5.2, the performance metrics for each programming language with respect to precision, recall and F1-score are given for XGBoost, and the confusion matrix is in Fig. 5.2. Note that the accuracy of training XGboost using textual information decreased by about 3% compared to the training using both the textual information and its code

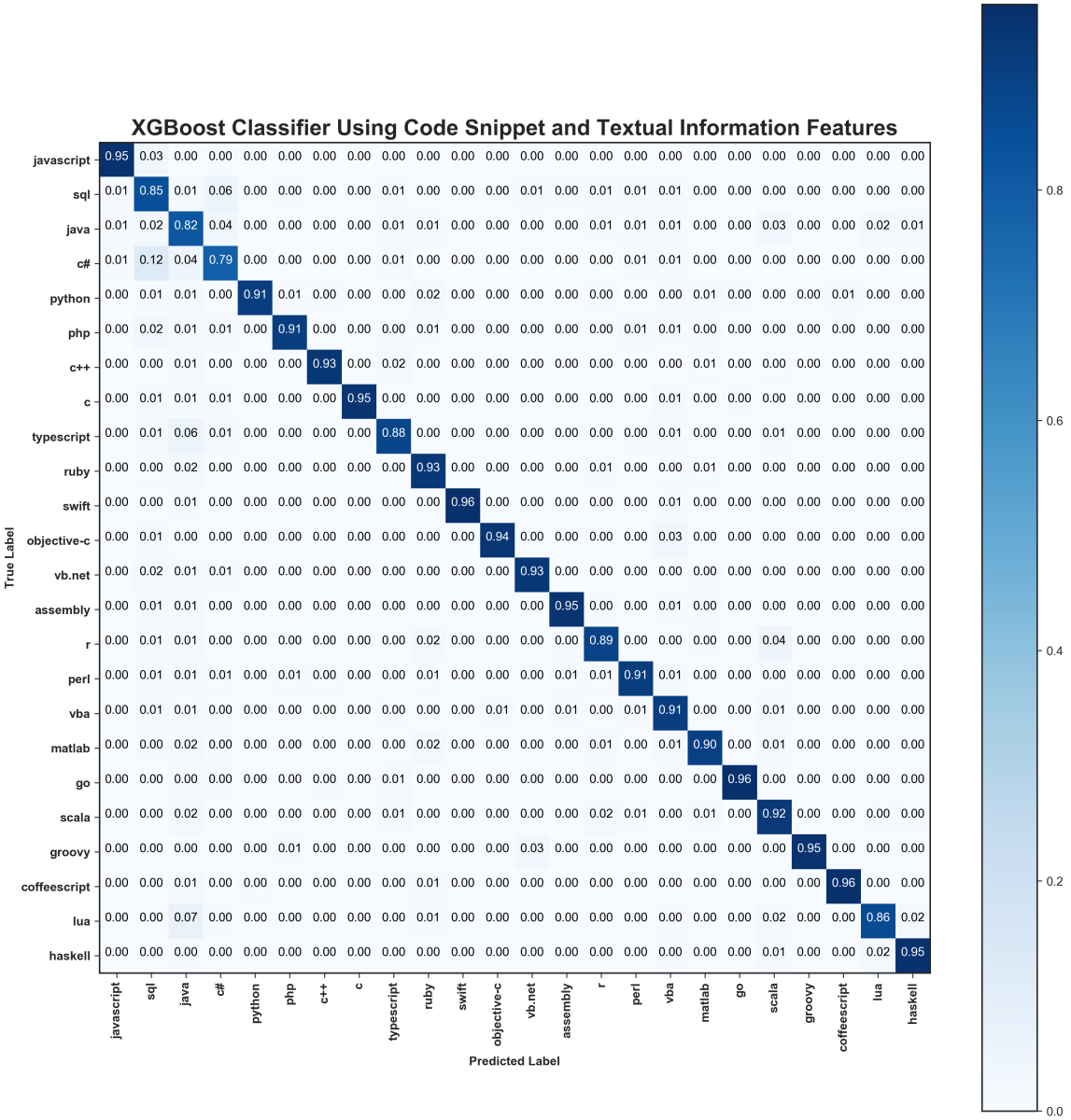


Figure 5.1: Confusion matrix for the XGBoost classifier trained on code snippet and textual information features. The diagonal represents the percentage of the programming language that was correctly predicted.

snippet. The top performing languages based on the F1-score are Swift (0.94), Go (0.94), C (0.93), CoffeeScript (0.93), Haskell (0.92), Groovy (0.91) Assembly (0.91) and JavaScript (0.91). Note further that the F1-scores of most of the programming languages in the table decreased by approximately 2% with a few exceptions (such as

C++, C#, Java and VBA). It should be further noted that the languages Swift, Go, CoffeeScript, Haskell and Assembly have a high F1-score and performed very well in both the (RQ1) and (RQ2).

Programming	**Precision**	**Recall**	**F1-score**
CoffeeScript	0.96	0.91	0.94
JavaScript	0.94	0.89	0.92
Swift	0.94	0.89	0.92
Go	0.95	0.89	0.92
Haskell	0.92	0.91	0.92
C	0.93	0.88	0.91
Objective-C	0.94	0.87	0.90
Assembly	0.92	0.87	0.89
Python	0.95	0.82	0.88
Groovy	0.95	0.82	0.88
C++	0.92	0.83	0.87
Ruby	0.86	0.88	0.87
R	0.88	0.82	0.85
Perl	0.88	0.81	0.84
MATLAB	0.88	0.80	0.84
Scala	0.80	0.90	0.84
TypeScript	0.86	0.80	0.83
Vb.Net	0.82	0.82	0.82
VBA	0.76	0.81	0.78
PHP	0.82	0.72	0.77
Lua	0.73	0.59	0.65
C#	0.65	0.61	0.63
SQL	0.42	0.75	0.54
Java	0.43	0.58	0.49

Table 5.2: Performance of XGBoost trained on textual information features.

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 average accuracy of 62.4%, and the average score for precision, recall and F1-score are 0.76, 0.62 and 0.67 respectively. In Table 5.4, the performance metrics for each programming language

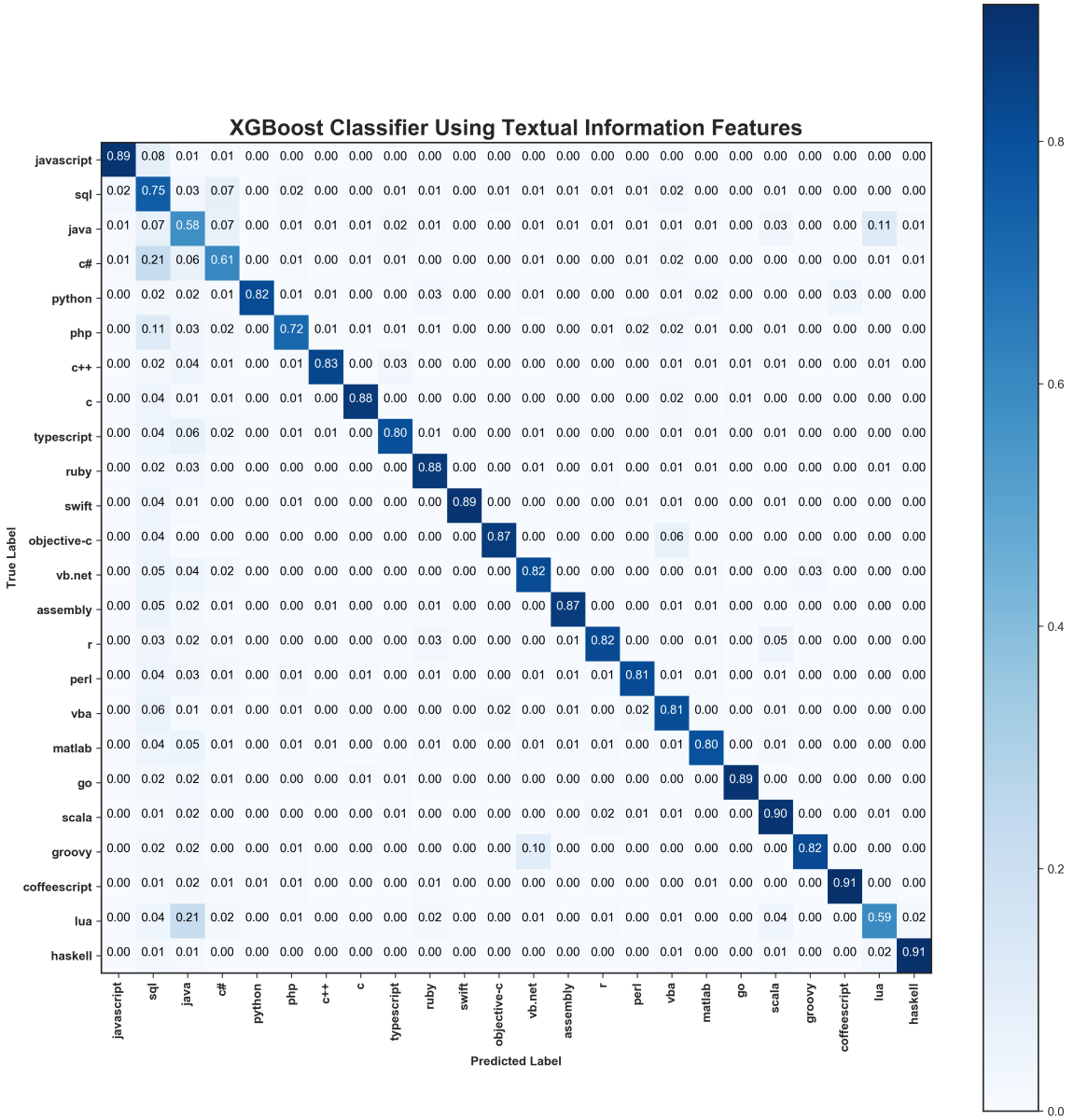


Figure 5.2: Confusion matrix for the XGboost classifier trained on textual information features. The diagonal represents the percentage of the programming language that was correctly predicted.

with respect to precision, recall and F1-score are given while using XGBoost. The programming languages Groovy (0.78), Go (0.79), CoffeeScript (0.79) still had a good F1-score. The F1-score of PHP in (RQ1) and (RQ2) is close to the average of all the programming languages; however, in (RQ3) the PHP language has the highest

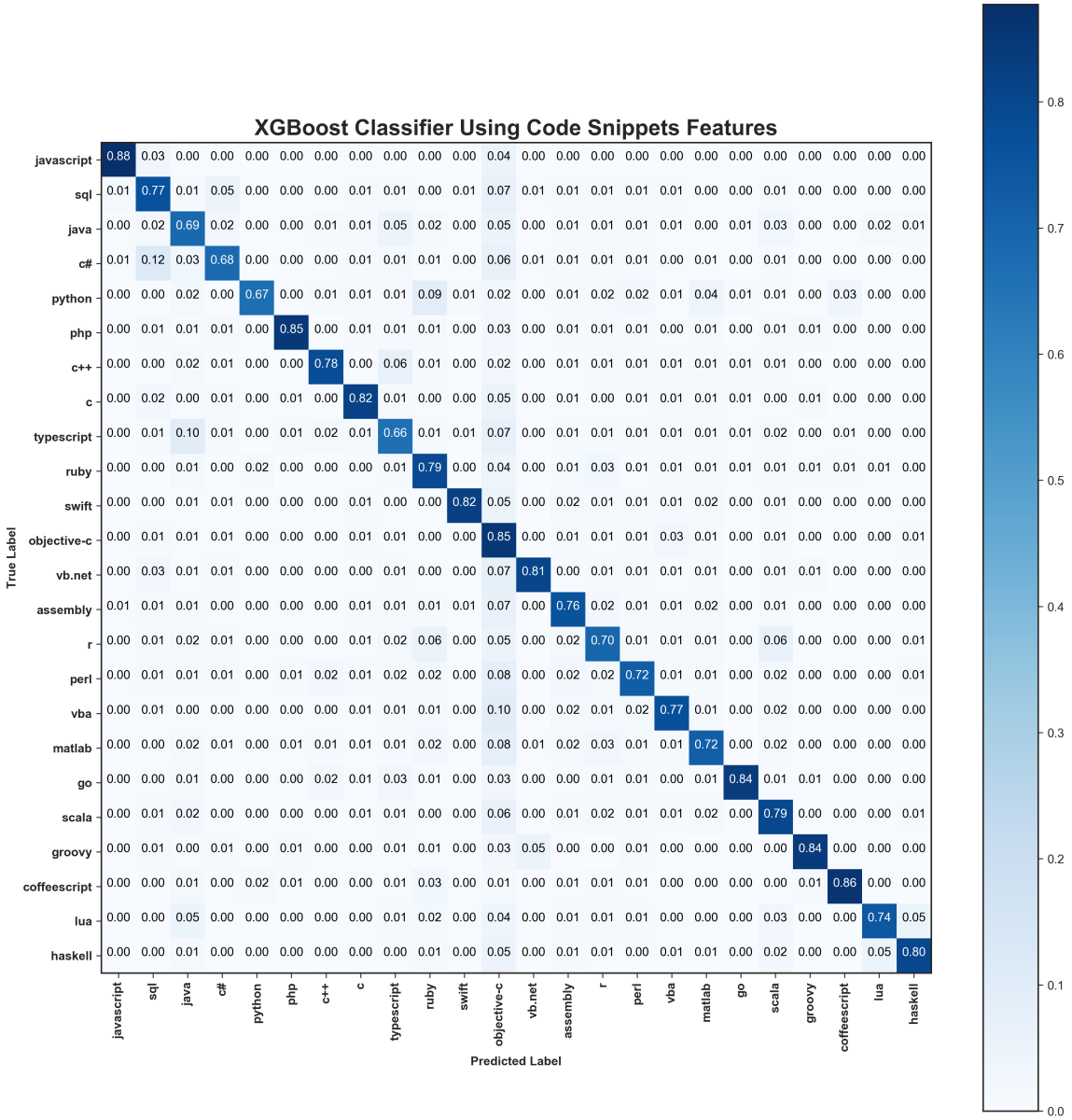


Figure 5.3: Confusion matrix for the XGboost classifier trained on code snippet features. The diagonal represents the percentage of the programming language that was correctly predicted.

F1-score. Objective-C has the worst F1-score and precision (0.25, 0.15); but the percentage of recall is extremely high (0.86). For some reason, XGBoost frequently misclassified snippets as Objective-C. Fig 5.3 shows the confusion matrix for the XGBoost classifier. Table 5.3 shows how the accuracy improves as the minimum size

of the code snippet in the dataset is increased from 10 to 100 characters.

The Minimum Characters	Accuracy	Precision	Recall	F1-score
More than 10	77.7%	0.79	0.77	0.78
More than 25	79.1%	0.80	0.97	0.79
More than 50	81.7%	0.82	0.81	0.81
More than 75	83.1%	0.83	0.83	0.83
More than 100	84.7%	0.85	0.84	0.84

Table 5.3: Effect of the minimum number of characters in a code snippet on the accuracy

Programming	Precision	Recall	F1-score
JavaScript	0.94	0.88	0.91
CoffeeScript	0.92	0.86	0.89
PHP	0.91	0.85	0.88
Go	0.92	0.84	0.87
Groovy	0.91	0.84	0.87
Swift	0.91	0.82	0.86
C	0.86	0.82	0.84
Vb.Net	0.89	0.81	0.84
Haskell	0.87	0.80	0.83
C++	0.86	0.78	0.82
VBA	0.82	0.77	0.80
Lua	0.87	0.74	0.80
Assembly	0.76	0.76	0.76
Python	0.85	0.67	0.75
Ruby	0.72	0.79	0.75
MATLAB	0.79	0.72	0.75
Scala	0.71	0.79	0.75
SQL	0.70	0.77	0.73
C#	0.78	0.68	0.73
Perl	0.75	0.72	0.73
R	0.72	0.70	0.71
TypeScript	0.68	0.66	0.67
Java	0.64	0.69	0.66
Objective-C	0.42	0.85	0.56

Table 5.4: Performance of XGBoost trained on code snippet features.

The comparison between the results of textual information dataset and code snippet dataset shows a huge difference in accuracy of 19% (in average). On the other

hand, using the combination of textual information and code snippet increases the accuracy only by 3% compared to using only the textual information.

5.2 Random Forest Classifier

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

The Random Forest Classifier achieves an accuracy of 86.3% and the average of precision, recall and F1-score are 0.87, 0.86 and 0.86 respectively, Table 5.5 shows the details for each programming language and the confusion matrix is shown in Fig. 5.4. Overall, this classifier achieves 5 % less accuracy than XGBoost classifier. Swift, Go and Groovy had the highest F1-score in both classifiers, but in Random forest classifier their F1-score dropped by about 2%. Java still had the worst F1-score among the programming languages, in XGBoost Classifier its F1-score was 0.75 and in this classifier decreased by 10%, to 0.65.

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

Random Forest still achieves lower accuracy compared to XGBoost Classifier. The accuracy dropped from 86.3% in (RQ1) to 74.5% in (RQ2). Table 5.6 shows the performance and the confusion matrix is in Fig 5.6. The average precision, recall and F1-score are 0.76, 0.74 and 0.75. When Random Forest classifier is used, the F1-score for CoffeeScript in (RQ1) was the sixth highest score (0.94) while in (RQ2) it was the second highest (.90), only a drop by 4%. Interestingly, when XGBoost classifier is used, CoffeeScript had the fourth highest F1-score in (RQ1) (0.79) and in (RQ2) its F1-score was the highest (0.94). That is, predicting CoffeeScript from textual information provides a better F1 score than combining textual information and code snippets. The F1-score for PHP was more than the average of all the programming languages in (RQ1), however in (RQ2) its F1-score was one of the worst among all

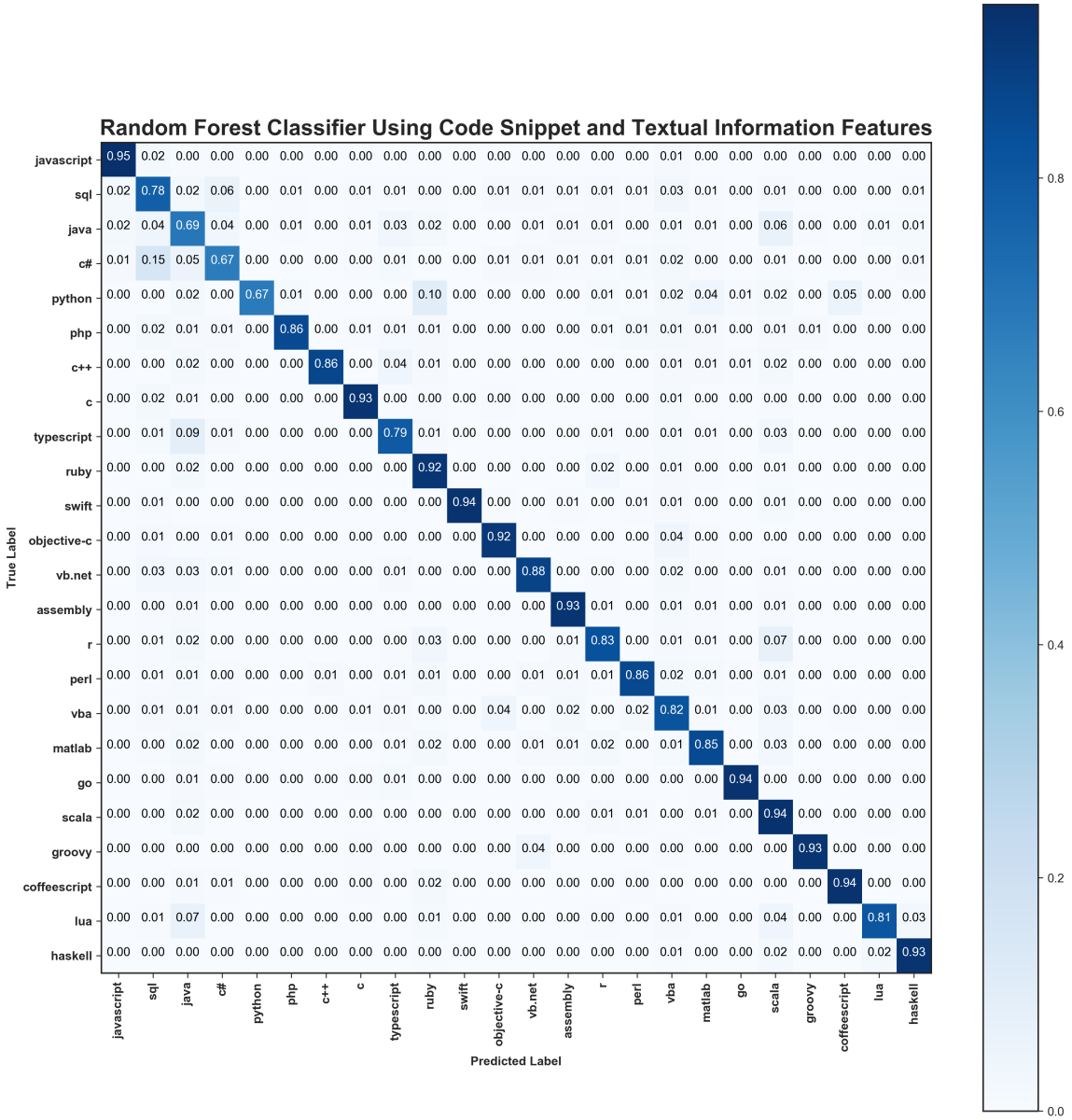


Figure 5.4: Confusion matrix for the Random Forest classifier trained on code snippet and textual information features. The diagonal represents the percentage of the programming language that was correctly predicted.

programming languages.

The recall of SQL (0.61) is higher than precision (0.34) and F1-score (0.44). The classifier predicts SQL correctly; however, other programming language are predicted as SQL. This is because SQL is embedded with many programming languages.

Programming	Precision	Recall	F1-score
Swift	0.96	0.94	0.95
Go	0.96	0.94	0.95
Groovy	0.97	0.93	0.95
JavaScript	0.92	0.95	0.94
C	0.94	0.93	0.94
CoffeeScript	0.95	0.94	0.94
Haskell	0.93	0.93	0.93
C++	0.98	0.86	0.92
Assembly	0.91	0.93	0.92
Objective-C	0.90	0.92	0.91
PHP	0.95	0.86	0.90
Vb.Net	0.89	0.88	0.89
Perl	0.89	0.86	0.87
Lua	0.94	0.81	0.87
Ruby	0.81	0.92	0.86
MATLAB	0.87	0.85	0.86
R	0.86	0.83	0.85
TypeScript	0.82	0.79	0.81
Python	0.99	0.67	0.80
Scala	0.70	0.94	0.80
VBA	0.75	0.82	0.78
SQL	0.69	0.78	0.73
C#	0.78	0.67	0.72
Java	0.61	0.69	0.65

Table 5.5: Performance of RFC trained on textual information and code snippet features.

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

Overall, the classifier achieved an accuracy of 70.1% in (RQ3). The average precision, recall and F1-score are 0.72, 0.70 and 0.70 respectively.

PHP language had the second highest F1-score in (RQ3) while in (RQ1) it was around the average among all the programming languages, and in (RQ2) it had a poor F1-score. This is shown in Table 5.7. It means that code snippets significantly help to predict the PHP language. Also, the F1-scores for JavaScript were higher in (RQ1)

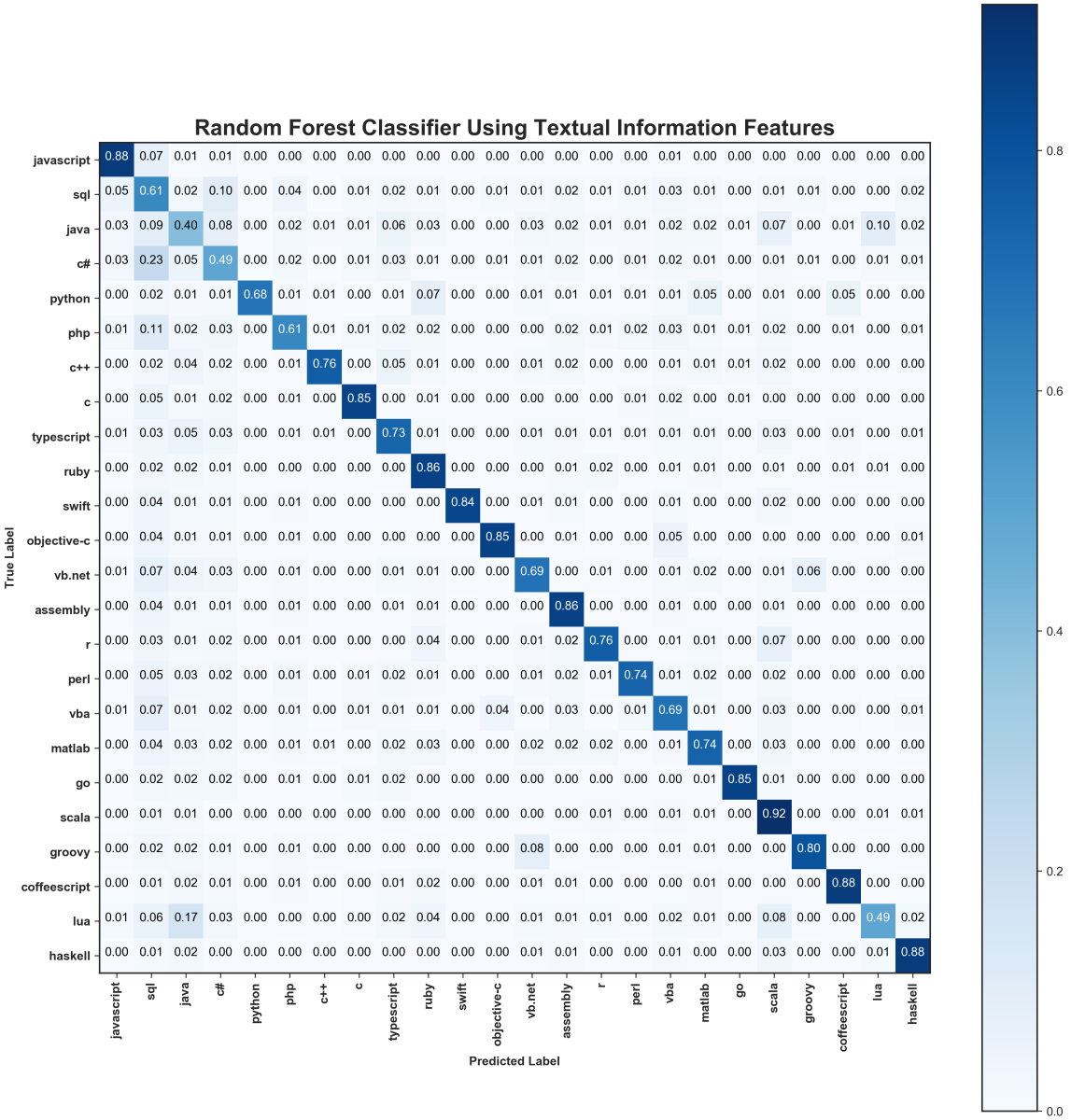


Figure 5.5: Confusion matrix for the RandomForest classifier trained on textual information features. The diagonal represents the percentage of the programming language that was correctly predicted.

and (RQ3) than (RQ2) because the codes snippet feature is more important than textual information feature to predict JavaScript. In (RQ1) and (RQ2), F1-score for Python were close to the average of all programming languages, but in (RQ3) it has the worst score (0.39). This is because so many Python code snippets were predicted

Programming	Precision	Recall	F1-score
Go	0.94	0.85	0.90
CoffeeScript	0.92	0.88	0.90
Swift	0.95	0.84	0.89
Objective-C	0.91	0.85	0.88
C	0.89	0.85	0.87
Haskell	0.87	0.88	0.87
JavaScript	0.82	0.88	0.85
Groovy	0.88	0.8	0.84
C++	0.92	0.76	0.83
Assembly	0.78	0.86	0.82
R	0.86	0.76	0.81
Python	0.97	0.68	0.80
Ruby	0.74	0.86	0.80
Perl	0.86	0.74	0.80
MATLAB	0.78	0.74	0.76
Scala	0.64	0.92	0.75
TypeScript	0.70	0.73	0.72
Vb.Net	0.73	0.69	0.71
VBA	0.70	0.69	0.69
PHP	0.75	0.61	0.68
Lua	0.75	0.49	0.60
C#	0.48	0.49	0.49
SQL	0.34	0.61	0.44
Java	0.38	0.40	0.39

Table 5.6: Performance of RFC trained on textual information features.

as Ruby (0.23) and CoffeeScript (0.12).

The syntax and semantics of TypeScript are very similar to Java as TypeScript was invented for programmers who had Java background. In our classifier, (0.11) of code snippet from TypeScript were predicted as Java. Moreover, (0.09) code snippets from Java were predicted as TypeScript showing that these two programming languages can be confused between each other.

JavaScript had the highest precision, recall and F1-score of 0.94, 0.81 and 0.87 respectively. The precision of Objective-C was the worst, because many code snippets were wrongly predicted as Objective-C as shown in the confusion matrix in Fig 5.6. The programming languages of some of these code snippets are extremely hard to

identify [1][2][3][4]. These snippets might be only contain a name of method, a name of class, a name of library, declaration, expression etc. They have extremely few features.

As a final remark, both classifiers (RQ1) and (RQ2) provide a better performance than (RQ3) as machine learning models learn better from textual information features compared to the code snippet alone.

Programming	**Precision**	**Recall**	**F1-score**
JavaScript	0.94	0.81	0.87
PHP	0.89	0.79	0.84
Groovy	0.86	0.8	0.83
CoffeeScript	0.84	0.78	0.81
Swift	0.85	0.74	0.80
Go	0.81	0.80	0.80
Vb.Net	0.81	0.74	0.78
Haskell	0.82	0.73	0.77
C	0.74	0.75	0.75
C++	0.83	0.65	0.73
Assembly	0.70	0.71	0.71
Lua	0.76	0.67	0.71
VBA	0.73	0.67	0.70
MATLAB	0.79	0.63	0.70
Ruby	0.62	0.78	0.69
Scala	0.62	0.78	0.69
SQL	0.64	0.71	0.67
Perl	0.67	0.64	0.65
C#	0.67	0.62	0.64
R	0.65	0.62	0.63
Java	0.56	0.57	0.57
TypeScript	0.54	0.60	0.57
Objective-C	0.36	0.73	0.48
Python	0.80	0.26	0.39

Table 5.7: Performance of RFC trained on code snippet features.

Table 5.8 summarizes the results in comparison to [4] as, to the best of our knowledge, it is the only previous work in the literature that tackles the problem of pre-

[1]https://stackoverflow.com/questions/855360/
[2]https://stackoverflow.com/questions/942772/
[3]https://stackoverflow.com/questions/9986404/
[4]https://stackoverflow.com/questions/2115227/

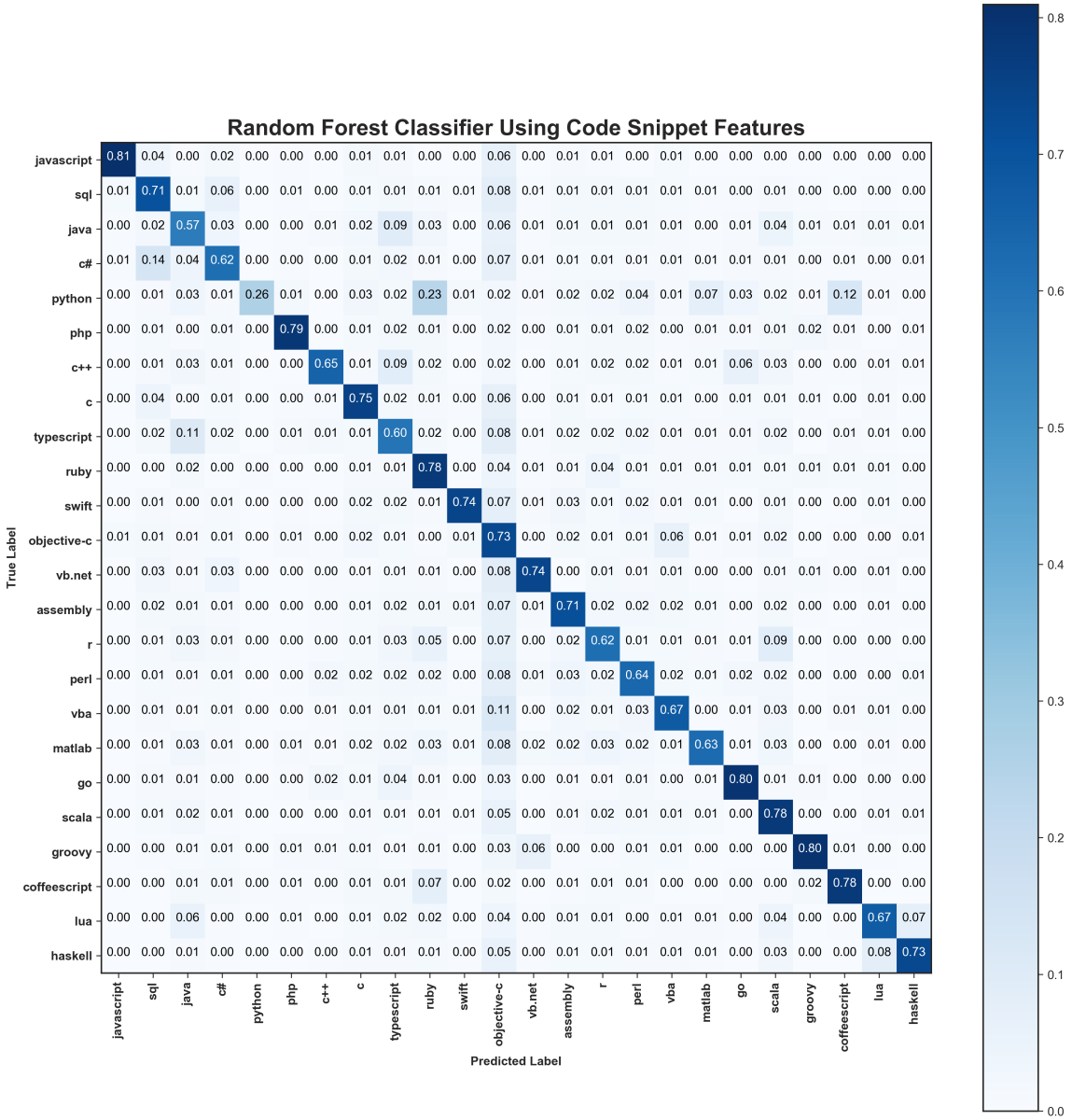


Figure 5.6: Confusion matrix for the Random Forest classifier trained on code snippet features. The diagonal represents the percentage of the programming language that was correctly predicted.

dicting programming language tags for Stack Overflow questions.

Table 5.8 shows that the results with the proposed classifiers are much better than with Baquero [4]. In this study, two advanced machine learning algorithms were used (XGboost and Random Forest) Baquero employs Support Vector Machine (SVM).

The dependency parsing and extraction of entity names for textual information in (RQ1) and (RQ2) using Spacy is the main reason for the significant improvement compared to Baquero. Further, the dataset was ten times larger than that used in [4].

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
Baquero [4] code snippets	44.6%	0.45	0.44	0.44
Baquero [4] textual information	60.8%	0.68	0.60	0.60
XGBoost (RQ1)	91.1%	0.91	0.91	0.91
XGBoost (RQ2)	81.1%	0.83	0.81	0.81
XGBoost (RQ3)	77.7%	0.79	0.77	0.78
Random Forest (RQ1)	86.3%	0.87	0.86	0.86
Random Forest (RQ2)	74.5%	0.76	0.74	0.75
Random Forest (RQ3)	70.1%	0.72	0.70	0.70

Table 5.8: A comparison of the classifier in Baquero [4] and the proposed classifiers.

Chapter 6

Discussion and Threats to Validity

6.1 Discussion

The most important observation in the previous section is that for the research question (RQ1), XGBoost achieves a high accuracy of 91.1%, while for (RQ2) and (RQ3), it only achieves an accuracy of 81.1% and 77.7% respectively. These observations highlight the importance of using a combination of textual information and code snippets in predicting tags in comparison to textual information or code snippets only. As the textual information for different programming language is very different, it is easier for a machine learning classifier to learn from textual information features rather than code snippet features. In some cases, Stack Overflow posts contain a very small code snippet making it extremely hard to identify its language as many programming languages share the same syntax.

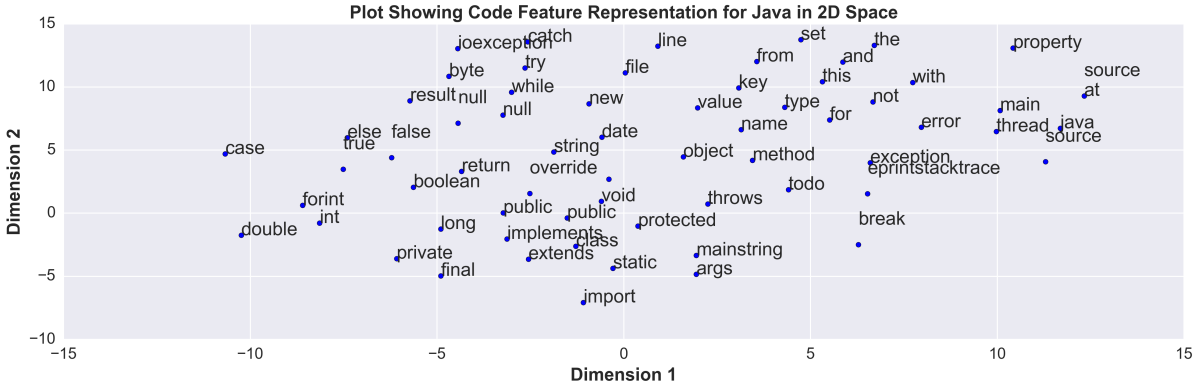
Dependency parsing and extracting of entity names using a Neural Network (NN) through Spacy reduced the noise and improved the extract important features from Stack Overflow questions. This is likely the main reason for the significant improvement in performance compared to previous approaches in the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

The analysis of the feature space for 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

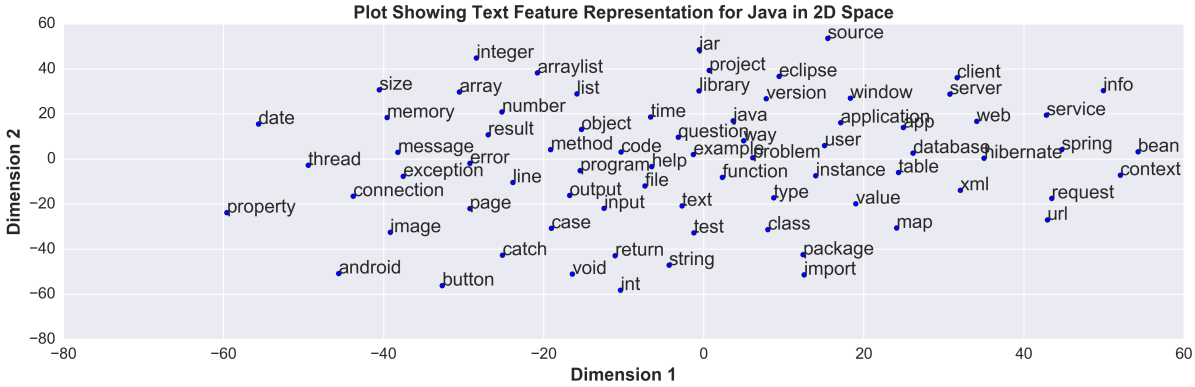
based features were visualized for Haskell, words such as ‘GHC’, ‘GHCI’, ‘Yesod’ and ‘Monad’ were obtained. ‘GHC’ and ‘GHCI’ are compilers for Haskell, ‘Yesod’ is a web-based framework and ‘Monad’ is a functional programming paradigm (Haskell is a functional programming language). Most of the top performing languages have a small feature space (vocabulary) as compared to more popular languages such as Java, VBA 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.

The Word2Vec models were trained in two different datasets, the code snippets and textual information, to visualize their features. Fig. 6.1 shows the features for Java and Fig. 6.2 shows the features for SQL. There were more than 200 features extracted using Word2Vec for each language, but for clarity, only about 70 features are shown. In the vector space, features which are used in the same context are close. For example, in the Java code snippets (Fig. 6.1), ‘public’,‘class’,‘implements’ and ‘extends’ are all close to one another in the vector space since they are typically used together in a line of code. Further, Fig. 6.2 shows that in SQL code snippets features such as ‘foreign’, ‘primary’, ‘key’, ‘constraint’ and ‘reference’ are all used in the context of setting relationships between tables.

The smaller cosine distance between features does not necessarily mean they are used in the same line of code but are used in the same context, e.g. declaring relationships in tables. This is because in Stack Overflow, developers describe their issues by including code based identifiers and features along with text not just in code blocks. Another critical observation from the Java textual information features is that ‘Android’ and ‘spring’ (a framework for web and desktop applications) are located at opposite sides of the plot (maximum cosine distance). This means that an Android developer rarely uses the spring framework for development. Conversely, ‘spring’ is located close to ‘web’, ‘service’, ‘url’, ‘request’ and ‘xml’, which indicates that spring framework is frequently used in web development in Java.



(a) Java code snippet features.



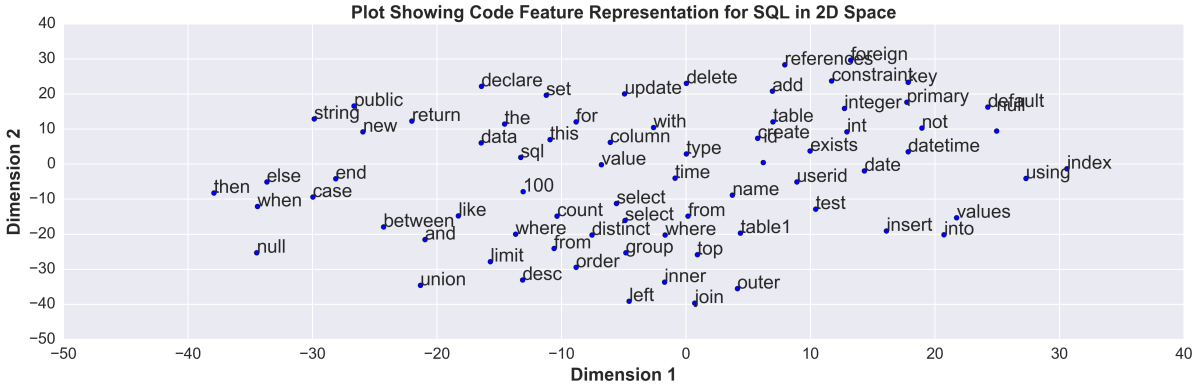
(b) Java textual information features.

Figure 6.1: Code snippet and textual information features of Java represented in two dimensions after using T-SNE on a trained Word2Vec model.

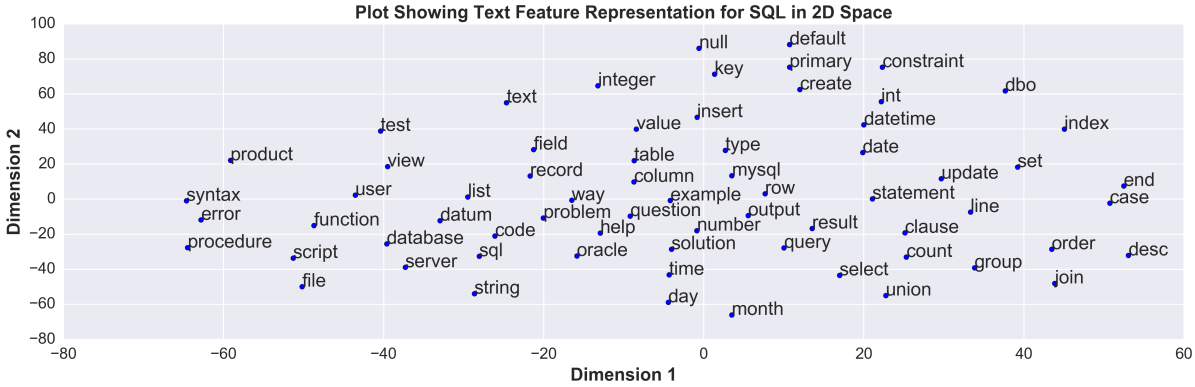
6.2 The Features

During the training stage, the machine learning algorithm extracts features from each programming language to help the model learn how to predict a programming language of a new post. The proposed models learn from the top $10,000$ features for each programming language to achieve a better accuracy. In this thesis, we only show and discuss the top 50 features as it is not possible to list $10,000$ features for 24 programming languages. The features are shown in Tables 6.1, 6.2, 6.3, 6.4 and 6.5.

Some features are very common among programming languages. For example, 'class' is one of the top features in 14 programming languages, JavaScript, Java, C#, Python, PHP, C++, TypeScript, Ruby, Swift, Vb.Net, Scala, Groovy, CoffeeScript



(a) SQL code snippet features.



(b) SQL textual information features.

Figure 6.2: Code snippet and text information features of SQL represented in two dimensions using T-SNE on a trained Word2Vec model.

and R. Also, a feature 'string' is a common keyword that is used to declare a variable type string. The keyword appears in top 50 features for all programming languages apart from JavaScript, SQL, Python and R. This is because there is no type declaration for a new variable in JavaScript, R and Python. Programming language syntax terms such as 'if', 'is', 'for', 'of', 'to', 'ture', 'in', 'the' and 'then' appear in the top 50 features of many programming languages.

The F1-score of JavaScript is the highest in (RQ3) among programming languages. It is the only programming language for web development compared to other 23 programming languages. Table 6.1 shows the top features for JavaScript and we can see that it has many unique features.

Programming Language	Top Features
JavaScript	alert, body, class, click, content, data, div, document, else, false, for, form, function, getelementbyid, height, href, html, http, id, if, img, input, is, javascript, jquery, js, length, li, name, new, onclick, option, return, script, span, src, style, td, text, the, this, title, to, tr, true, type, value, var, width, window
SQL	as, begin, by, case, count, create, date, datetime, dbo, declare, default, desc, end, for, from, group, id, if, in, inner, insert, int, into, is, join, key, left, name, not, null, on, or, order, primary, select, set, sql, sum, t1, table, the, then, to, type, update, value, values, varchar, when, where
Java	add, apache, at, catch, class, com, exception, false, file, final, for, http, id, if, import, in, int, is, java, javax, lang, list, main, method, name, new, null, object, of, org, out, println, private, property, public, return, source, static, string, sun, system, the, this, to, true, try, type, value, void, xml
C#	add, asp, at, bool, byte, class, console, data, else, false, foo, for, foreach, from, get, id, if, in, int, is, item, list, name, new, null, object, of, private, public, return, select, sender, server, set, static, string, system, text, the, this, to, tostring, true, type, using, value, var, void, where, writeline
Python	__init__, and, class, data, db, def, django, error, file, foo, for, from, http, id, if, import, in, is, lib, line, models, module, name, none, not, object, of, open, os, packages, path, print, py, python, python2, request, return, self, site, sys, test, text, the, this, to, true, type, user, usr, value

Table 6.1: The top 50 features for each programming language.

Programming Language	Top Features
PHP	_post, and, array, as, br, class, com, content, data, div, echo, else, file, for, from, function, html, http, id, if, in, input, is, name, new, null, of, option, php, public, query, result, return, row, select, string, td, test, text, the, this, title, to, true, type, url, user, value, where, www
C++	and, bool, boost, char, class, const, cout, cpp, data, double, else, end, endl, error, file, foo, for, function, if, in, include, int, is, it, main, name, new, null, of, operator, private, public, return, size, std, string, struct, template, test, the, this, to, type, typename, unsigned, using, value, vector, virtual, void
C	and, argc, argv, array, break, buffer, case, char, const, count, data, define, double, else, error, exit, file, for, from, function, if, in, include, int, is, list, long, main, malloc, name, next, node, null, of, printf, return, size, sizeof, stdio, string, struct, temp, the, this, to, typedef, unsigned, value, void, while
TypeScript	angular, angular2, any, app, class, component, console, constructor, core, data, div, error, export, false, from, function, html, http, id, if, import, interface, is, js, json, let, log, module, name, new, ng, node_modules, number, path, private, public, require, return, router, script, src, string, the, this, to, true, ts, type, value, var
Ruby	activerecord, and, base, bin, class, com, def, div, do, each, end, error, file, foo, for, from, gem, gems, html, http, id, if, in, include, is, lib, library, local, name, new, nil, params, post, puts, rails, rake, rb, require, ruby, rubygems, self, string, test, the, to, true, type, user, users, usr

Table 6.2: The top 50 features for each programming language.

Programming Language	Top Features
Swift	anyobject, array, as, bool, case, cell, class, data, else, error, false, for, frame, func, height, iboutlet, if, image, import, in, indexpath, init, int, is, let, name, nil, not, nsstring, of, override, println, return, self, sender, size, string, super, swift, tableview, text, the, to, true, uikit, value, var, view, viewdidload, width
Objective-C	alloc, array, bool, cell, count, data, delegate, else, end, error, for, frame, id, if, import, in, indexpath, init, int, interface, is, name, nil, no, nonatomic, nsarray, nslog, nsmutablearray, nsstring, objectatindex, of, property, release, retain, return, selector, self, size, string, stringwithformat, super, tableview, text, the, to, uiimage, uiview, view, void, yes
Vb.Net	add, and, as, asp, byval, class, data, dim, end, false, for, from, function, get, handles, id, if, in, integer, is, item, me, name, new, next, not, nothing, object, of, private, property, public, return, runat, select, sender, server, set, string, sub, system, text, the, then, to, tostring, true, try, type, value
Perl	and, at, bin, data, die, error, file, foo, for, foreach, from, hash, html, http, id, if, in, is, lib, line, my, name, new, not, of, open, or, perl, pl, pm, print, return, self, shift, strict, string, sub, test, text, the, this, to, txt, type, use, usr, value, warnings, while, xml
VBA	activecell, activesheet, add, and, application, as, cell, cells, count, data, dim, each, else, end, error, false, for, function, if, in, integer, is, long, me, msgbox, name, new, next, not, nothing, of, offset, private, range, row, rows, select, selection, set, sheets, string, sub, text, the, then, to, true, value, with, worksheets

Table 6.3: The top 50 features for each programming language.

Programming Language	Top Features
MATLAB	all, and, at, data, disp, double, else, end, error, fid, figure, file, find, for, from, function, get, handles, if, ii, image, in, input, int, is, length, line, matlab, matrix, max, name, nan, not, obj, of, on, pi, plot, rand, set, sin, size, string, sum, the, this, time, to, value, zeros
Go	body, byte, chan, com, data, err, error, file, fmt, for, func, get, github, go, http, id, if, import, in, int, interface, is, json, log, main, make, map, name, net, new, nil, of, os, package, printf, println, range, request, return, src, string, struct, test, the, time, to, type, user, value, var
Scala	apply, array, at, bar, case, class, collection, com, def, error, extends, foo, for, id, if, implicit, import, in, int, is, java, lang, list, main, map, match, name, new, not, object, of, option, org, println, sbt, scala, seq, some, string, test, the, this, to, trait, type, user, val, value, var, with
Groovy	apache, at, class, codehaus, com, def, error, false, file, for, gradle, grails, groovy, http, id, if, import, in, info, invoke, is, it, java, lang, list, method, name, new, null, of, org, params, println, public, return, run, runtime, static, string, test, the, this, to, true, type, user, util, value, version, xml
CoffeeScript	app, at, backbone, class, coffee, console, data, div, else, end, err, error, false, for, function, get, html, http, id, if, in, is, jquery, js, json, lib, log, model, module, name, new, node_modules, object, on, options, render, require, return, scope, script, test, text, the, this, to, true, type, url, user, value

Table 6.4: The top 50 features for each programming language.

Programming Language	Top Features
Lua	addeventlistener, and, data, display, do, else, elseif, end, error, event, false, file, for, function, id, if, in, insert, io, is, local, love, lua, math, name, new, nil, no, not, of, or, phase, player, png, print, random, require, return, scene, self, string, table, test, text, the, then, time, to, true, value
Haskell	as, bool, cabal, char, control, data, deriving, do, else, eq, error, expression, foo, for, from, ghc, hs, if, import, in, instance, int, integer, io, is, just, let, list, main, map, maybe, monad, not, nothing, num, of, print, putstrln, return, show, string, test, text, the, then, to, true, type, where, xs
Assembly	21h, a0, add, ah, al, and, ax, bx, byte, call, cmp, code, cx, data, db, dword, dx, eax, ebp, ebx, ecx, edi, edx, end, esi, esp, for, if, in, inc, int, is, jmp, loop, main, mov, movl, number, of, pop, push, r0, ret, si, sp, string, text, the, to, v0
R	aes, and, as, by, class, col, csv, data, date, df, error, factor, false, file, for, frame, from, function, ggplot, id, if, in, is, length, library, list, matrix, mean, na, name, names, not, null, of, paste, plot, print, read, rep, rnorm, sep, structure, table, test, the, time, to, true, type, value

Table 6.5: The top 50 features for each programming language.

6.3 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.

Internal validity: After the datasets were extracted, dependency parsing was used to select entity names so as to include only the most relevant code snippet and text features. The use of dependency parsing can result in the loss of critical vocabulary and might affect the results. However, we manually analyzed the vocabulary before and after the dependency parsing to ensure that information related to the languages was not lost. Further, selecting additional features such as lines of code and programming paradigm could have improved our results but was not considered.

External validity: The focus of this thesis was to obtain a classifier for predicting languages due to the lack of open source tools for this task. Stack Overflow was used in this study as the data source but other sources such as GitHub repositories were not explored. Therefore, no conclusions can be made about the results with other sources of code snippets and text on programming languages. Furthermore, some common programming languages such as Cobol and Pascal were not considered in this study.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This work tackled the important problem of predicting programming languages from code snippets and textual information. In particular, it focused on predicting the programming language of Stack Overflow questions. The results showed that training and testing the classifier by combining the textual information and code snippet achieves the highest accuracy of 91.1%. Other experiments using either textual information or code snippets, achieved accuracies of 81.1% and 77.7% respectively. This implies that information from textual features is easier for a machine learning model to learn compared to information from code snippet features. The results also showed that it is possible to identify the programming language of a snippet with just a few lines of source code. We believe that this classifier could be applied in other scenarios such as code search engines and snippet management tool.

7.2 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 it 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 will vary depending on the programming language. Therefore, it is important that the vocabulary for each programming language is captured, understood and separated. It is suggested that a CNN be used with Word2Vec for NLP related tasks. CNNs have been shown to be very effective in document classification tasks compared to most simple classifiers.

The proposed model was trained and evaluated using only Stack Overflow questions. In the future, this model will be evaluated using programming blog posts, library documentation and bug repositories. This would help us in understanding how general the model is.

Appendix A

Additional Information

In this appendix, the top 50 features for the textual information discussed in (RQ1) are shown in Tables A.1, A.2, A.3, A.4, A.5 and A.6. These features help the machine learning model to learn and predict the programming languages in the discussion forum. This model achieves a higher accuracy compared to models that learn the features from code snippets. This means that the discussion of a particular programming language is unique. Also, these features show the main topics that are discussed in Stack Overflow for each programming language.

Programming Language	***Top Features***
JavaScript	ajax, array, asp, box, browser, button, class, code, content, css, div, document, dom, element, error, event, example, file, firefox, form, function, help, html, image, input, javascript, jquery, json, line, link, list, method, object, page, problem, question, script, server, solution, string, table, tag, text, time, url, user, value, way, web, window
SQL	access, case, clause, code, column, count, database, date, datum, error, example, field, function, group, help, index, insert, join, key, list, mysql, number, oracle, order, performance, problem, procedure, pron, query, question, record, result, row, select, server, set, solution, sql, statement, string, syntax, table, time, type, update, user, value, view, way, would
Java	app, application, array, case, class, client, code, database, eclipse, error, example, exception, file, hibernate, instance, interface, jar, java, jsp, line, list, method, number, object, output, page, problem, program, project, question, server, service, servlet, solution, source, spring, string, table, test, text, thread, time, type, user, value, version, way, web, window, xml
C#	app, application, array, asp, case, class, code, control, custom, database, edit, error, event, example, exception, file, form, function, instance, interface, line, linq, list, method, net, object, page, problem, project, property, query, question, reference, server, service, solution, sql, string, table, test, text, thread, time, type, user, value, way, web, window, xml
Python	app, application, case, class, code, command, database, dictionary, directory, django, edit, error, example, field, file, form, function, help, html, instance, library, line, list, loop, method, model, module, number, object, output, page, problem, process, program, py, python, question, script, server, solution, string, table, test, text, time, type, user, value, way, window

Table A.1: The top 50 textual information features for each programming language.

Programming Language	***Top Features***
C++	application, array, base, boost, case, char, class, code, compiler, const, constructor, cpp, dll, edit, error, example, file, function, header, int, library, line, list, map, member, memory, method, number, object, operator, output, pointer, problem, program, project, question, reference, size, solution, std, string, template, thread, time, type, value, variable, vector, way, window
C	address, application, array, bit, buffer, case, char, character, code, command, compiler, edit, error, example, file, function, gcc, header, help, input, int, integer, library, line, linux, list, loop, malloc, memory, number, output, pointer, problem, process, program, question, server, size, source, string, struct, structure, thread, time, type, user, value, variable, way, window
TypeScript	angular, app, application, array, class, code, compiler, component, constructor, controller, definition, directive, error, example, file, folder, function, html, import, interface, issue, javascript, js, json, library, line, method, module, node, object, page, problem, project, property, question, reference, service, string, studio, template, test, time, ts, tsconfig, type, typescript, value, variable, version, way
Ruby	activerecord, app, application, array, block, class, code, command, controller, database, error, example, file, form, gem, hash, help, html, instance, lib, line, list, method, model, module, number, object, output, page, post, problem, question, rail, rake, rb, ruby, script, server, string, table, test, text, time, user, value, variable, version, view, way, xml

Table A.2: The top 50 textual information features for each programming language.

Programming Language	***Top Features***
Swift	app, apple, application, array, beta, button, cell, class, code, compiler, controller, custom, dictionary, error, example, file, function, help, image, init, int, ios, issue, json, line, method, object, objective, problem, project, property, protocol, question, screen, self, storyboard, string, swift, table, tableview, text, time, type, user, value, variable, view, viewcontroller, way, xcode
Objective-C	app, application, array, button, cell, class, cocoa, code, controller, core, custom, datum, delegate, error, example, file, function, help, image, instance, interface, iphone, line, memory, method, nsmutablearray, number, object, objective, problem, program, project, property, question, screen, self, string, table, text, time, type, uitableview, uiview, user, value, variable, view, way, window, xcode
Vb.Net	app, application, array, asp, button, class, code, column, control, database, error, event, example, exception, file, form, function, help, line, linq, list, message, method, net, object, page, problem, program, project, property, query, question, row, server, service, sql, string, table, text, textbox, time, type, user, value, vb, visual, way, web, window, xml
Perl	array, case, cgi, class, code, command, cpan, datum, directory, error, example, expression, file, function, hash, help, html, input, line, list, loop, method, module, number, object, output, page, perl, problem, process, program, question, regex, result, script, server, solution, string, subroutine, test, text, time, use, user, value, variable, version, way, window, xml

Table A.3: The top 50 textual information features for each programming language.

Programming Language	***Top Features***
MATLAB	array, axis, case, cell, class, code, column, command, datum, element, error, example, figure, file, function, gui, help, image, index, input, line, loop, matlab, matrix, method, number, object, order, output, plot, point, problem, program, question, result, row, script, set, size, solution, string, structure, text, time, user, value, variable, vector, way, window
Go	api, app, application, array, case, channel, client, code, command, directory, error, example, field, file, function, go, golang, google, html, http, int, interface, json, library, line, map, method, object, output, package, pointer, problem, program, question, request, response, result, server, slice, string, struct, structure, template, test, time, type, user, value, variable, way
Scala	actor, application, array, case, class, code, collection, compiler, constructor, error, example, file, function, idea, instance, int, java, json, library, lift, line, list, map, match, message, method, object, option, order, parameter, pattern, play, problem, project, question, result, sbt, scala, solution, string, test, time, trait, type, user, val, value, version, way, xml
Groovy	app, application, build, case, class, closure, code, command, controller, domain, error, example, exception, field, file, gradle, grail, groovy, help, java, jenkin, json, lang, line, list, map, method, object, output, plugin, problem, project, property, question, request, response, result, script, service, soapui, string, task, test, time, type, user, value, version, way, xml

Table A.4: The top 50 textual information features for each programming language.

Programming Language	***Top Features***
Lua	array, attempt, button, case, class, code, command, corona, error, event, example, file, function, game, help, image, index, input, issue, key, line, list, loop, lua, method, module, number, object, output, path, pattern, player, problem, program, question, result, scene, screen, script, sdk, server, string, table, text, time, torch, user, value, variable, way
Haskell	cabal, case, class, code, datum, definition, element, error, example, expression, file, function, ghc, ghci, haskell, help, hs, idea, implementation, input, instance, int, library, line, list, map, memory, module, monad, number, order, output, package, pattern, point, problem, program, question, result, solution, state, string, test, text, time, tree, type, value, version, way
Assembly	address, arm, array, asm, assembler, assembly, bit, byte, case, character, code, command, compiler, eax, error, example, file, function, gcc, help, input, instruction, int, intel, language, line, linux, loop, masm, memory, mip, mode, mov, nasm, number, output, pointer, problem, program, question, register, result, stack, string, syntax, time, user, value, variable, way
R	advance, axis, case, code, column, command, data, dataframe, dataset, date, datum, error, example, factor, file, format, frame, function, ggplot, graph, group, help, length, line, list, loop, matrix, model, number, object, order, output, package, plot, point, problem, question, result, row, script, series, set, solution, table, text, time, value, variable, vector, way

Table A.5: The top 50 textual information features for each programming language.

Programming Language	***Top Features***
PHP	application, array, class, code, content, database, date, error, example, field, file, form, function, help, html, image, index, input, line, method, mysql, number, object, output, page, php, post, problem, query, question, regex, request, result, script, server, session, site, string, table, text, time, type, url, user, value, variable, way, web, xml, zend
VBA	access, application, array, box, button, cell, code, column, database, date, datum, document, error, example, excel, field, file, form, format, formula, function, help, line, list, loop, macro, method, number, object, problem, query, question, range, row, script, sheet, string, sub, table, text, time, type, user, value, variable, vba, way, word, workbook, worksheet
CoffeeScript	ajax, app, application, array, backbone, button, callback, class, code, coffee, coffeescript, collection, console, controller, element, error, event, example, file, form, function, help, html, issue, javascript, jquery, js, json, line, list, method, model, module, node, object, page, problem, project, question, request, script, server, template, test, time, user, value, variable, view, way

Table A.6: The top 50 textual information features for each programming language.

Bibliography

[1] Developer survey results 2017. `https://insights.stackoverflow.com/survey/2017#technology`. Accessed: 2018-04-30.

[2] Tiobe programming community index. `https://www.tiobe.com`. Accessed: 2018-04-30.

[3] M. Ahasanuzzaman, M. Asaduzzaman, C. Roy, and K. Schneider. "Mining duplicate questions of stack overflow". In *IEEE Working Conference on Mining Software Repositories*, pages 402–412, 2016.

[4] J. Baquero, J. Camargo, Restrepo-Calle, F. Aponte, and F. González. "Predicting the programming language: Extracting knowledge from stack overflow posts". In *Colombian Conference on Computing*, pages 199–210. Springer, 2017.

[5] A. Barua, S Thomas, and A. Hassan. "What are developers talking about? An analysis of topics and trends in stack overflow". *Empirical Software Engineering*, 19(3):619–654, 2014.

[6] R. Berk. *"Statistical Learning From a Regression Perspective"*. Springer, 2016.

[7] S. Bird and E. Loper. "NLTK: The natural language toolkit". In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, page 31, 2004.

[8] L. Breiman. "Random forests". *Machine learning*, 45(1):5–32, 2001.

[9] T. Chen and C. Guestrin. "XGBoost: A scalable tree boosting system". In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[10] D. Correa and A. Sureka. "Chaff from the wheat: Characterization and modeling of deleted questions on stack overflow". In *ACM International Conference on World wide web*, pages 631–642, 2014.

[11] R. Holmes, R. Walker, and G. Murphy. "Strathcona example recommendation tool". In *ACM SIGSOFT Software Engineering Notes*, pages 237–240, 2005.

[12] J. Kennedy, V. Dam, and V. Zaytsev. "Software language identification with natural language classifiers". In *IEEE Software Analysis, Evolution, and Reengineering Conference*, pages 624–628, 2016.

[13] J. Khasnabish, M. Sodhi, J. Deshmukh, and G. Srinivasaraghavan. "Detecting programming language from source code using Bayesian learning techniques". In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 513–522, 2014.

[14] N. Lester, L. Feldman, and F Prado Martín. "You can take a noun out of syntax. . . : Syntactic similarity effects in lexical priming". In *Proceedings of the Annual Meeting of the Cognitive Science Society*, pages 2537–42, 2017.

[15] L. Maaten and G. Hinton. "Visualizing data using t-sne". *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[16] C. McMillan, M. Grechanik, D. Poshyvanyk, Q. Xie, and C. Fu. "Portfolio: Finding relevant functions and their usage". In *ACM International Conference on Software Engineering*, pages 111–120, 2011.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient estimation of word representations in vector space". *International Conference on Learning Representations*, 2013.

[18] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. "Distributed representations of words and phrases and their compositionality". In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[19] S. Nasehi, J. Sillito, F. Maurer, and C. Burns. "What makes a good code example?: A study of programming Q&A in stack overflow". In *IEEE International Conference on Software Maintenance*, pages 25–34, 2012.

[20] F. *et al.* Pedregosa. "Scikit-learn: Machine learning in python". *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] G. Pinto, F. Castor, and Y. Liu. "Mining questions about software energy consumption". In *IEEE Working Conference on Mining Software Repositories*, pages 22–31, 2014.

[22] R. Rehurek and P. Sojka. "Gensim–python framework for vector space modelling". *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 2011.

[23] V. Rekha, N. Divya, and P. Bagavathi. "A hybrid auto-tagging system for stackoverflow forum questions". In *ACM International Conference on Interdisciplinary Advances in Applied Computing*, page 56, 2014.

[24] C. Rosen and E. Shihab. "What are mobile developers asking about? A large scale study using stack overflow". *Empirical Software Engineering*, 21(3):1192–1223, 2016.

[25] Saha Saha, A. K and R. Schneider. "A discriminative model approach for suggesting tags automatically for stack overflow questions". In *IEEE Working Conference on Mining Software Repositories*, pages 73–76, 2013.

[26] C. Stanley and M. Byrne. "Predicting tags for stack overflow posts". In *International Conference on Cognitive Modeling*, 2013.

[27] C. Treude, O. Barzilay, and M. A. Storey. "How do programmers ask and answer questions on the web?: NIER track". In *IEEE International Conference on Software Engineering*, pages 804–807, 2011.